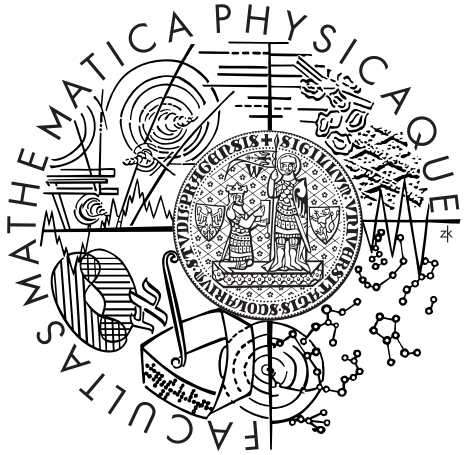
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁRSKA PRÁCA



Juraj Hámorník

Rozpoznávanie písaných znakov v reálnom čase

Katedra teoretické informatiky a matematické logiky

Vedúci bakaláskej práce:	Mgr. Daniel Toropila
Študijny program:	Informatika
Študijny obor:	Správa počítačových systémov

Praha 2012

Ďakujem pánovi Mgr.Danielovi Toropilovi za odborné vedenie tejto práce, za rady a za čas, ktorý mi počas vypracovávania tejto práce venoval.

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne a výhradne s použitím citovaných prameňov, literatúry a ďalších odborných zdrojov.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Zb. Autorského zákona v platnom znení, najmä skutočnosť, že Univerzita Karlova v Praze má právno na uzavretie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe dňa 2. augusta 2012 Podpis autora

Názov práce: Rozpoznávanie písaných znakov v reálnom čase

Autor: Juraj Hámorník

Katedra: Katedra teoretické informatiky a matematické logiky

Vedúci bakalárskej práce: Mgr. Daniel Toropila

Abstrakt: Zariadenia Tablet PC sú v dnešnej dobe čoraz obľúbenejšie. Keďže neobsahujú hardwarovú klávesnicu, užívateľ do nich vkladá text dvomi hlavnými spôsobmi: virtuálnou klávesnicou, kde sú vkladané jednotlivé znaky postupne a analýzou rukou napísaného textu, kde sú vkladané celé slová. Nami vytvorená metóda kombinuje po znakoch zadávaného textu s analýzou rukou písaných znakov. Rukou napísaný znak, ktorý môže pozostávať viacerých ťahov je prevedený na sekvencie čísiel. Na základe týchto sekvencii a užívateľom naučených znakov vyhodnotíme napísaný znak. Keď sme porovnali vstavané metódy pre zadávanie textu na operačnom systéme Windows 8 zistili sme, že pre skúsenejších užívatelov a isté jazyky môže byť naša metóda prínosná.

Klúčové slová: rozpoznávanie písaných znakov, vstup pre tablety, analýza vzorov

Title: Real-time Recognition of Typed Letters

Author: Juraj Hámorník

Department:

Department of Theoretical Computer Science and Mathematical Logic

Supervisor: Mgr. Daniel Toropila

Abstract: Tablet PC devices are becoming more and more popular these days. These devices rarely have a hardware keyboard. Users input a text in two major ways. First one is a virtual keyboard, where users type each character individually and second one is handwrite recognition, where users write the whole words. Our method combines chars input method and handwriting. Hand written character that can contain multiple strokes, is converted into sequences of numbers. Based on this sequence we evaluate a written character. When we compare the default input method on operating system Windows 8 we can figure, that for advanced users , and for some languages our method could be a challange.

Keywords: handwrite recognition , patern recognition , tablet imput

Obsah

Úvod

Ešte pred pár rokmi bola forma zaznamenávania a predávania informácií rukou písaným písmom. Predávanie informácií vo forme rukou napísaného textu prežila stovky rokov. Dnes je ale postupne vytláčaná počítačmi a klávesnicou. V poslednej dobe sú čím ďalej, tým viac obľúbené zariadenia tablet PC, ktoré nahrádzajú klasické počítače a hardwarovú klávesnicu nemajú. Prinášajú možnosti pre zadávanie textu opäť rukou.

Cieľom tejto práce bolo vyvinúť funkčnú implementáciu fungujúcu na platforme tablet PC postupu prevedenia rukou písaného textu do počítača. Zameriame sa na cieľovú skupinu používajúcu túto platformu. Vstupný rukopis, ktorý budeme príjmať, bude mať jasne separované znaky. Tieto potom prevedieme na základe smerov písania, nie výzoru na text. Rukopis je veľmi individuálna záležitosť. Nami vyvinutá metóda toto akceptuje a poskytuje jednoduchý a transparentný spôsob, ako si sám užívateľ do veľkej miery môže naučiť program na jeho písmo. Nekladú sa žiadne obmedzenia na tvar. Znak A môžeme písať napríklad ako domček.

Zakladá myšlienka prevodu sa opiera o spôsoby, akými sa učíme písať. Na základe, akým smerom pohybujeme perom pri písaní vkladaného znaku vytvoríme sekvencie čísel, ktoré dostatočne popisujú tento znak. Vďaka tejto sekvencii a dátam zaznamenaných pri učení zistíme, o aký znak sa jedná. Idea vkladania textu postupne pomocou samostatných znakov vychádza zo starej metódy Graffity [2] dostupnej na zariadeniach Palm.

V kapitole 1 sa pozrieme na vývoj platformy tablet PC a pokúsime sa vyvrátiť fámu, že je to vynález tohto desaťročia. V kapitole 2 sa pozrieme veľmi stručne na to, čo je kaligrafia, a aký prínos pre našu metódu má. Konkurenčné spôsoby zadávania textu zhrnieme v kapitole 3. Chybám v nich sa pokúsime vyhnúť a prednosti použiť.

Samotné riešenie problému je popísané v kapitole 4. Postupne prejdeme postup na rozloženie vloženého znaku na tri jednoduché sekvencie čísiel. Následne ich spracujeme na znak. Musíme vyriešiť problém z chybami, ktoré vzniknú pri písaní rukou, ako aj problémy s nejednoznačnosťou napísaného znaku.

Softwarová implementácia pre operačný systém Windows je opísaná v kapitole 5. Kapitola je rozdelená na užívateľskú príručku, ktorá pomôže pri používaní program a na programátorskú príručku. Programátorská príručka opisuje vnútornú štruktúru programu a netriviálne implementačné riešenia. Taktiež vysvetľuje zvolený jazyk, prístup k písaniu programu a jeho konečný výzor.

Keďže cieľom je vytvoriť použiteľný program, v kapitole 6 si najprv predstavíme konkurenčné programy a následne testy, ktorým ich vystavíme. Na základe výsledkov testov vyvodíme dôsledky. V závere práce zhrnieme prínos nášho riešenia a načrtneme ďalšie možné smerovanie a možný vývoj tejto práce.

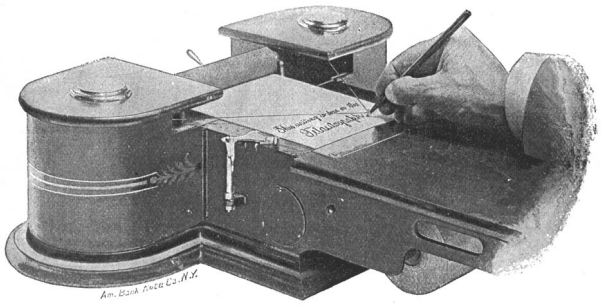
1. Zariadenia tablet PC v priebehu času

Pojem tablet sa v posledných rokoch dostal do povedomia aj širšej verejnosti. Teraz tak označujeme tenké zariadenie s veľkým displejom, spravidla bez klávesnice, ktoré sa ovláda prstami, výnimočne špeciálnym perom. Vyznačuje sa vysokou výdržou na batérie a dobrou konektivitou. Používa sa najmä na konzumovanie obsahu, ako prezeranie www stránok, počúvanie hudby, sledovanie filmov a hranie hier.

Koncept prenosného zariadenia, ktoré by bolo ľahké, prenosné a pre užívateľa intuitívne na ovládanie nie je nové. Zariadenia podobné tabletom sa vyskytujú vo vedecko-fantastických filmoch od 60. rokov minulého storočia. V seriáli Startrek používali postavy prenosné počítače ovládané stylusom (špeciálnym perom). Známejšie účinkovanie tabletu bolo vo filme Stevena Kubricka Vesmírna odysea 2001, kde posádka vesmírnej lode sledovala bezdrôtovo správy na tenkom plochom displeji veľkosti stránky papiera.

1.1 Pred-počítačová éra

Prvé základy boli položené už v roku 1888, kedy bol udelený patent na zariadenie nazývané teleaotographer [1], ktoré slúžilo na prenos rukou písaného textu a najmä na prenos podpisov a obrázkov po telegrafickom vedení. Zariadenie (obr. 1.1) pozostávalo z pera, ktoré bolo pripevnené dvomi nitkami na potenciometre. Každý pohyb pera bol rozložený na horizontálnu a vertikálnu zložku pohybu a tá bola prenášaná káblom k totožnému zariadeniu, na ktorom sa hýbalo pero po papieri rovnako, ako na prvom zariadení.



Obr. 1.1: Nákres teleautographera

1.2 Počítačová éra

O tablet sa pokúšalo veľa firiem, niektoré viac, niektoré menej úspešne a niektoré Najprv sa vôbec nepoužívalo priame ovládanie dotykmi prstov, ale na ovládanie sa používalo špeciálne pero (stylus).

GRIDPad

Prvým predávaným zaradením bol GRIDPad (obr. 1.2) od spoločnosti GRID predstavené v roku 1989. Jednalo sa o drahé (2400 $) zariadenie osadené 10-Mhz procesorom značky Intel. Vzhľadom a proporčne sa veľmi približovalo dnešným tabletom. Ovládalo sa perom a nemalo zabudovanú žiadnu podporu pre rozpoznávanie rukou písaného textu.

Obr. 1.2: Zariadenie GRIDPad

GO PenPoint

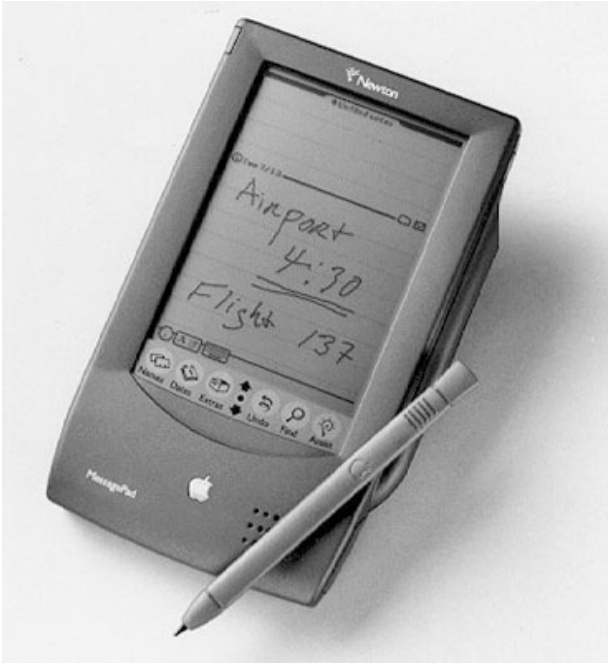
Zariadenie GO PenPoint bolo ideologickým nasledovníkom okrem toho, že toto zariadenie bolo spúšťačom pre Pen Windows a malo zaujímavo poňaté prostredie. Zariadenie bolo poňaté ako poznámkový blok, takže namiesto vytvorenia nového dokumentu sa vytvorila nová stránka v bloku.

Microsoft Windows for Pen Computing

Odpoveďou Microsoftu na tablety bola platforma Microsoft Windows for Pen Computing. Systém bol postavený nad operačným systémom Windows 3.1 s upraveným ovládaním pre pero. Ovládanie bolo do veľkej miery postavené nad gestami, čo bolo jeho silnou, ale aj slabou stránkou. Ovládanie bolo rýchle, ale bolo veľmi ťažké odhaliť gestá a následne si ich zapamätať. Pôvodný neúspech nasledovala verzia 2, ktorá bola postavená nad Windows 95 a nemala tiež veľký úspech. Platforma doplatila na nezáujem ako výrobcov, tak aj zákazníkov a najmä na vtedajší nedostatok výpočtového výkonu na komplexnú analýzu rukou písaného textu.

Apple Newton

Veľký podiel má aj spoločnosť Apple [3]. Ešte pred iPadom vyvíjala spoločnosť Apple modelovú radu Apple Newton(obr. 1.3). Pre toto zariadenie bol prvý krát použitý názov PDA (Personal digital asistent). Toto zariadenie trpelo viacerými nedostatkami. Bolo príliš veľké, takže sa zle držalo, malo náchylnú obrazovku na poškodenie a nie efektívne navrhnuté zadávanie rukou písaného textu. Pôvodný software na analýzu rukopisu bol slovne orientovaný s databázou len 10 000 slov. V novších verziách už bola táto chyba opravená, ale vďaka poškodenej povesti zo začiatku a zlej konektivite s PC, bola výroba týchto tabletov pozastavená.



Obr. 1.3: Zariadenie Apple Newton

Palm

Od roku 1996 vyrábala firma Palm software a neskoršie aj hardware. Jej systém sa vyznačoval unikátnym spôsobom zadávania textu, takzvané Graffity [2] (na základoch ktorého je postavená aj naša metóda) a jednotlačidlovú synchronizáciu. Zariadenia s cenou aj pod 300 $ (obr 1.4) boli prvými komerčne úspešnými tabletmi. Spoločnosť dominovala trhu s vreckovými počítačmi dlhé roky až do roku 2005. Aktuálne bola spoločnosť prevzatá firmou HP, ktorá uvoľnila operačný systém webOs (postavený z časti na pôvodnom Palm OS) ako open source.

Obr. 1.4: Zariadenie Palm Pilot 1000

Microsoft Pocket PC

Odpoveďou Microsoftu bolo uvedenie systému Microsoft Pocket PC. Zaujímavou bola až tretia generácia s prepracovaným ovládaním a pokročilým zadávaním textu. Zatiaľ čo sa firma Palm zamerala na masy, Microsoft Pocket PC sa viac zameriaval na korporátnych zákazníkov. Softwarovo sa nejednalo o nič revolučné. Pozbierali a vylepšili technológie z predchádzajúcich zariadení, napríklad rozpoznávanie písma bolo odvodené z programu ParaGraphs Caligrapher, ktorý bol použitý v prvých zariadeniach Apple Newton. Z platformy voľne vychádza aj Windows for Tablet PC. Zaujímavé je, že pred predstavením Apple iPhonu plafroma Microsoft Pocket PC prakticky ovládala trh so smartfónmi (telefónmi s operačným systémom).

1.3 Post-Pc éra

Dnešná doba sa niekedy označuje ako post-PC éra práve pre enormný nárast popularity tabletov. Predpokladá sa, že predaj tabletov by mal už v roku 2013 predbehnúť predaj klasických počítačov.

Modernú dobu tabletov začala spoločnosť Apple v roku 2010 vydaním svojho tabletu Apple iPad, ktorý je poháňaný mobilným systémom iOs. Označuje sa za revolučné zariadenie, ktoré nanovo zadefinovalo tvar tabletu. Zanedlho ho nasledovalo mnoho výrobcov, ktorí predstavili tablety postavené na mobilnom operačnom systéme Android.

Obe platformy sú postavené nad operačným systémom primárne založenom pre mobilné telefóny. Ohľadom zadávania rukou písaného textu sú pre nás obe platformy nezaujímavé. V základe poskytujú zadávanie textu len pomocou virtuálnej

klávesnice. Vďaka použitej technológií digitalizéra (zariadenia na zaznamenania dotyku) primárne určeného na ovládanie prstami nie sú veľmi vhodné na zadávanie textu písaním rukou. Existujú špeciálne dotykové perá, ktoré napodobňujú prst. Pre svoju malú hrúbku sú presnejšie, ale užívateľ nesmie mať na zariadení položenú ruku, čo robí písanie nesmierne namáhavým a nepohodlným. Jeden z najlepších kaligraferov sa vyjadril na otázku písania prstami,

> Písaním prstami vznikne rukopis, ktorý by som sa hanbil zavesiť na dvere chladničky[5].

1.4 Microsoft Tablet PC

Pri opise histórie tabletov sme sa zámerne vyhýbali jednej platforme, a to platforme Microsoft Tablet PC [5]. Platforma bola predstavená v roku 2002 s upraveným operačným systémom Windows XP for Tablet PC a donedávna nebola veľmi komerčne úspešná. Našla si uplatnenie najmä v oblastiach, kde je ovládanie perom a písanie virtuálnym atramentom veľmi dôležité, napríklad v zdravotníctve alebo v školstve. Technológia Tablet PC bola už potom zahrnutá v nadchádzajúcich operačných systémoch Windows a to ako aj v operačnom systéme Windows Vista, tak v aktuálnom operačnom systéme Windows 7.

Tablet PC je podľa Microsoftu definovaný ako zariadenie, ktoré spĺňa nasledujúce vlastnosti [5] :

1. Zariadenie musí obsahovať digitalizér, ktorý musí detekovať pero aj keď je tesne nad povrchom, ale nedotýka sa ho. Presnosť digitalizéra musí byť minimálne 600 dpi (dots per inch) a musí poskytovať polohu pera aspoň 100 krát za minútu.

2. Zariadenie musí byť schopné prebrať sa z módu spánku do dvoch sekúnd (zariadenie nie je úplne vypnuté, napájaná zostáva pamäť a niektoré iné vnútorné obvody).

3. Zariadenie musí byť schopné zmeniť orientáciu obrazovky z rozloženia na výšku/na šírku bez reštartovania.

4. Zariadenie musí byť vybavené hardwarovým tlačítkom s funkciou ekvivalentnou trojkombinácii CTRL + ALT + DELETE. Prítomnosť tlačítka nie je nutná pre reštartovanie systému, ale daná kombinácia je potrebná pri bezpečnom prihlásení do domény.

Nadchádzajúci operačný systém Windows 8 je silne zameraný na platformu Tablet PC. Prináša nové užívateľské rozhranie Metro primárne určené na ovládanie prstami a dotykovými perami. Operačný systém Windows 8 so sebou prináša aj prvé počítače vyrábané priamo Microsoftom. Jedná sa o zariadenia Microsoft Surface PRO a RT. Tablet s prívlastkom PRO je vybavený aj stylusom.

2. Rýchly pohľad do kaligrafie

V krátkosti si predstavíme formu vstupu, teda písmo, konkrétnejšie veľké tlačené písmo. Kaligrafia (pochádzajúca z gréckeho kallos krása + grafos písanie) je voľne povedané umenie krasopisu. Zahrňuje jednak funkčné nápisy a ručne písané texty, ako aj výtvarné umenie, kde vizuálna stránka je uprednosťnovaná pred stránkou funkčnou.

2.1 Románska abeceda

Románska abeceda, ktorá je známa viac ako latinská, je najviac používaná abeceda na svete. Text, ktorý pravé čítate je napísaný touto abecedou. Ako jedna z mála abecied pozostáva z dvoch foriem znakov:

- malých písmen (a b c d e f g h i j k l m n o p q r s t u v w x y z)
- veľkých písmen (A B C D E F G H I J K L M N O P Q R S T U V W X Y Z)

Názov pochádza od starovekých Rimanov, ktorí ju používali skrátenú (neobsahovala písmená J, U a W) a najmä neobsahovala malé písmená. Rimania prevzali a upravili abecedu od Grékov. Tí ju prevzali od Feničanov, ktorí ju poskladali od národov na Strednom východe. Rimania sa postarali o jej rozšírenie do celej Európy a odtiaľ neskôr do celého sveta.[6]

2.2 Vplyv kaligrafie na rozpoznávania písma

Vďaka možnosti sledovať a zaznamenávať celú trajektóriu pera pri písaní, nie sme odkázaní len na vizuálnu stránku písania, tj. ako písmeno vyzerá, ale vieme aj to, ako sme sa k danému písmenu dopracovali. Už v základnej škole nás učili nie len ako majú písmena vyzerať, ale aj ako ich máme písať. Na obrázku (obr. 2.1) je návod ako písať rustikálne veľké písmená D, P a R. Na obrázku (obr. 2.1) môžeme vidieť, akými ťahmi pera sa dopracujeme k výsledku.

Obr. 2.1: Kaligrafický návod na písanie písmen D, P a R.

Písmo sa dá dekomponovať na niekoľko jednoduchých tvarov, ktorých unikátnou kombináciou vznikne výsledné písmeno. Ak vynecháme ozdobné prvky rustikálneho štýlu písania, jedná sa o esenciálnu myšlienku nášho spôsobu rozpoznávania písma. Nebudeme sa zameriavať na to, ako dané písmeno vyzerá, ale ako sme sa dopracovali k jeho tvaru .

3. Zadávanie textu do zariadení tablet

Pozrieme sa na formy zadávania textu pre zariadenia tablet. Existujú hlavne tri smery zadávania. Prvá je virtuálna klávesnica, ktorá nemá žiadnu spojitosť s ručne písaným textom. Ďalšie dve sú vkladanie textu po znakoch s upravenou znakovou sadou a vkladanie textu prirodzenou formou.

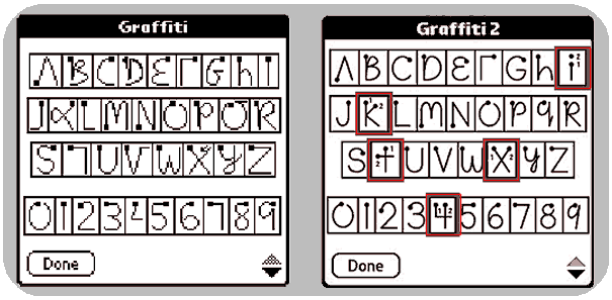
3.1 Zadávanie textu pomocou virtuálnej klávesnice

V dnešnej dobe najpopulárnejšia forma zadávania textu do zariadenia bez klávesnice je pomocou virtuálnej klávesnice. Koncept je veľmi jednoduchý. Na istej časti displeja je zobrazená virtuálna klávesnica. Užívateľ dotykom prsta na virtuálnu klávesu zadá požadovaný znak. Virtuálne klávesnice sa líšia najmä rozložením kláves od klasického rozloženia QWERTY (prípadne QWERTZ), cez exotické rozloženie DVORAK, až po rozdelenie klávesnice na 2 polovice pre jednoduchšie písanie palcami. Zaujímavou alternatívou je klávesnica typu SwiftKey, kde sa na požadované virtuálne klávesy priamo nekliká, ale sa len prstom prechádza cez písmená, ktoré chceme v slove použiť. Program na základe lexikálnej analýzy konkrétneho jazyka ponúkne množinu potenciálnych slov, z ktorých si užívateľ vyberie.

3.2 Zadávanie textu predom definovanou sadou znakov

V minulosti, keď výkon zariadení nebol dostatočný na komplexnejšiu analýzu rukopisu, vyvinula firma Xerox zjednodušenú znakovú sadu latinky postavenú na písme pre nevidiacich Moonotype [19]. Jednalo sa o celú abecedu, kde sa každé písmeno písalo unikátnym spôsobom a jedným ťahom. Písalo sa na predom určené miesto a znaky museli mať jednotnú veľkosť.

Tento spôsob prevzala firma Palm pod názvom Graffity. Následovník Graffity 2 [2], priniesol možnosť písania niektorých písmen prirodzenejšou formou viacerými ťahmi (obr.3.1). Táto metóda je po naučení a osvojení abecedy prekvapivo rýchla a presná. Nevýhodou je nulová personalizácia písma, takže nevyhovuje užívateľom, ktorí majú už zaužívané isté zlozvyky pri písaní. Unikátny je open source projekt Cellwriter[**?**]. Jedná sa o projekt pod záštitou Minnesotskej univerzity. Ide o spôsob zadávania jednotlivých znakov do mriežky. Program je možné naučiť spôsob písania každého znaku, no nemá žiadnu lexikálnu časť, takže má podporu pre viac jazykov. Jedná sa o veľmi prepracovaný a funkčný projekt, nanešťastie je dostupný len pre operačný systém Unix.



Obr. 3.1: Ukážka spôsobu písania pomocou systému Graffity

3.3 Zadávanie textu prirodzenou formou

Segmentácia a rozoznávanie znakov a slov alebo až celých viet je dobre preskúmaná oblasť a viacero komerčných implementácií je už nejaký čas dostupných [15]. Moderné komerčné systémy dokážu poskytnúť úspešnosť cez 90 percent, dokonca aj s neskúseným užívateľom bez žiadneho trénovania [16]. Aby toho komerčné systémy dosiahli, sú vyvíjané špeciálne len na jeden jazyk. Napríklad systém rozpoznávania písma, ktorý je dostupný v operačnom systéme Microsoft Windows, dokáže veľmi úspešné rozpoznávať ako aj tlačené písmená (latinku), tak aj šikmý rukopis. Na to používa opozdenie (systém čaká, či sa nepridá ďalší ťah, ak nie tak začne analýza). Súčasťou analýzy je analýza rukopisu pomocou neurónovej sieť a pokročilá lexikálna analýza [17]. Dokáže rozpoznávať len jazyky, pre ktoré je individuálne vyvinutá neurónová sieť a lexikálny model, takže sú to len majoritné jazyky, napríklad angličtina, nemčina, francúzština, ale aj čeština. Slovenčina nie je podporovaná. Ešte v operačnom systéme Windows 7 bola dostupná možnosť písať znaky postupne za sebou separátne do predom pripravených kolóniek podobne ako v metóde Cellwriter. Tento spôsob bol vhodný pre jazyky, ktoré neboli podporované. Naneštastie v novom operačnom systéme je táto metóda vynechaná.

4. Znakovo orientovaný spôsob vkladania textu

V predchádzajúcich kapitolách sme si priblížili vývoj platformy Tablet PC, ako aj metódy zadávania textu. Riešenie, ktoré si predstavíme, kombinuje prednosti predstavených systémov. Znakovo orientovaný spôsob je primárne určený na zariadenia bez displeja citlivého na dotyk rukou. V tejto práci predstavený prístup je orientovaný na vkladanie písmen latinskej abecedy. Ako si ďalej ukážeme, nebude náš prístup viazaný na jeden konkrétny jazyk, ale bude podporovať viac jazykov.

Riešenie je ideologicky najbližšie systému Graffity [2] od spoločnosti Palm. Práve absencia systému Graffity alebo jemu podobná metóda zadávania textu na platforme Tablet PC nás priviedla k tejto téme. Zadávanie textu budeme realizovať po jednotlivých znakoch. Tvar znaku bude nadefinovaný, ale na rozdiel od systému grafity bude mať užívateľ možnosť si každý znak predefinovať podľa jemu blízkemu štýlu písania. Pre každý znak si bude môcť užívateľ nadefinovať aj viac spôsobov písania.

Na rozdiel od systému pre vkladanie textu od Microsoftu nie je pri vkladaní slov (v našom prípade znakov) žiadne alebo len minimálne časové zdržanie. V našom riešení nie je žiadny lingvistický model, teda užívateľ nie je limitovaný nutnosťou písať slová v jednom konkrétnom jazyku.

Klávesnica je dobrý spôsob vkladania textu pri (multi) dotykových obrazovkách, kde môže užívateľ využívať prstoklady naučené z hardwarovej klávesnice, prípadne používať palce pri držaní zariadenia v rukách. Pri ovládaní stylusom sa nezachováva pohodlnosť a rýchlosť zadávania textu.

4.0.1 Zamietnuté metódy riešenia

Pri prevode rukou písaného písma sa skoro výhradne vždy používajú neurónové siete. Pri neurónových sieťach vzniká na základe učenia istá schopnosť abstrakcie pravidiel medzi vstupom a výstupom. Na základe takto vzniknutých pravidiel ich vieme aplikovať na akékoľvek vstupné údaje. Riešenie pomocou neurónových sietí sme nevybrali z viacerých dôvodov. Prvým dôvodom je nutnosť enormne veľkých dát na učenie, rádovo v desiatkach až stovkách vzorov na jeden znak. Druhým dôvodom je strata kontroly nad integritou uložených dát. Po naučení je veľmi náročné zmazať jeden konkrétny druh písania znaku. Zároveň sa neurónová sieť stáva neprehľadnou a nevieme jednoducho zistiť, na základe akých vlastností bolo aplikované dané pravidlo. Nami vytvorená metóda bude ľahko a rýchlo naučiteľná a budeme jasne vidieť, podľa čoho prevedieme rukou napísaný znak.

Riešenie

Toto riešenie pozostáva z niekoľkých krokov. Obecne sa dá povedať, že vstup v podobe užívateľom napísaných kriviek a úsečiek (rukou napísaný znak) prevediem na 2 sekvencie čísel. Jednu označíme ako vektorová sekvencia a druhú ako pozičná sekvencia. Podľa Markovovho modelu [8] ich analyzujeme a priradíme k nim odpovedajúci znak.

4.1 Vektorová sekvencia

Vektorovou sekvenciou nazveme sekvenciu čísel, ktorá nám bude udávať, akým smerom sme viedli ťah. Číselne teda zapíšeme popis vedenia ťahu ako je napríklad: zvislo hore (0) alebo šikmo dole sprava doľava (5). Tento spôsob kompresie krivky sa nazýva Freemenove kódovanie [14]. Nech máme sekvenciu N bodov $p_1..p_n$, ktorú nazveme krivka K(ťah).

$$K = \{p_i = (x_i, y_i | i = 1..n)\}$$

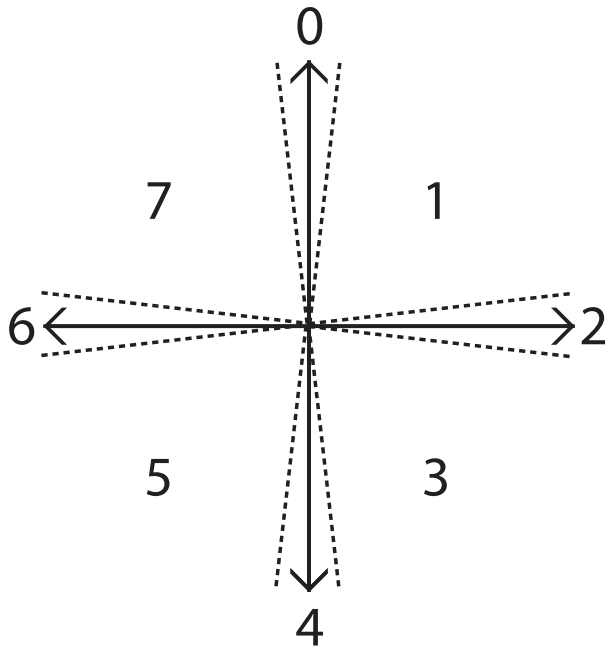
Freemanov reťazový kód F pozostáva z M vektorov, ktoré vytvoríme postupne z dvojíc bodov krivky K. Postupne pre každý bod $p_2..p_n \in K$ vytvoríme vektor tak, že p_{i-1} bude počiatočný bod vektoru a bod p_i konečný bod vektoru.
Inak povedané :

$$F = \{\vec{f}(C_i) | C_i = p_{i-1}\vec{}, p_i; p_{i-i}, p_i \in K\}$$

pričom každý vektor budeme reprezentovaný číslom $f = 0..7$. Číslo, ktoré reprezentuje vektor V získame na základe uhla α, ktorý zviera vektor V s osou Y. Celú os si rozdelíme do itervalov. Podla toho, do ktorého intervalu uhol α spadá, také číslo reprezentuje vektor V.

- $(0° - \zeta, 0° + \zeta) = 0$
- $[0° + \zeta, 90° - \zeta] = 1$
- $(90° - \zeta, 90° + \zeta) = 2$
- $[90° + \zeta, 180° - \zeta] = 3$
- $(180° - \zeta, 180° + \zeta) = 4$
- $[180° + \zeta, 180° - \zeta] = 5$
- $(270° - \zeta, 270° + \zeta) = 6$
- $[270° + \zeta, 360° - \zeta] = 7$

Odchýlka od osi ζ je užívateľsky nastaviteľná a zabezpečuje istú mieru voľnosti pri písaní, kde užívateľ nemusí napísať presne rovnú zvislú alebo vodorovnú čiaru, ale aj čiaru, ktorá je zhruba zvislá alebo vodorovná. Čísla $0, 2, 4, 6$ sú priradené pre smery hore, doprava, dole, doľava. Ostatné čísla sú priradené oblastiam medzi týmito hlavnými smermi. Defaultne je hodnota $\alpha = 7°$. Názorne môžeme vidieť rozdelenie na obrázku 4.1.



Obr. 4.1: Priradenie Freemenoveho kódu k vektoru

4.1.1 Detekcia rohov na priamke

V [9] bol prezentovaný objav, že útvar tvorený krivkami sa dá koncentrovať na dominantné body, ktoré majú vysokú krivosť (high curvative). Vzniklo veľa algoritmov na nájdenie takýchto bodov. Hlavné sú dva prístupy k tomuto problému. Prvý a nami použitý prístup používa na zistenie dominantných bodov priamy prístup cez analýzu uhlov alebo zisťovanie rohov. Druhý prístup získava tieto body postupne polygonálne lineárnou aproximáciou krivky. Takto nájdené dominantné body zodpovedajú skutočne dominantným bodom alebo zodpovedajú segmentu, s tvarom, ktorý hľadáme.
Použijeme dva podobné algoritmy. Najprv si vysvetlíme niektoré pojmy, ktoré oba algoritmy využívajú. Vstupom algoritmu je krivka K, tvorená bodmi
$K = \{p_i = (x_i, y_i | i = 1..n)\}$. Veľkosť rohu (rohovitosť) je vypočítaná na každý bod, pričom sú vybrané body na základe tohto výpočtu. Keď spracúvame bod p_i , algoritmy využívajú rôzne počty predchádzajúcich a nadchádzajúcich bodov, ako potencionálne ramená rohu pre p_i. Pre celé kladné číslo k predchádzajúci a nadchádzajúci $k - vektor$ pre bod pi definujeme ako

$$a_{ik} = (x_i - x_{i+k}, y_i - y_{i+k})$$

$$b_{ik} = (x_i - x_{i-k}, y_i - y_{i-k})$$

Algoritmus Rosenfeld and Johnston RJ73

Rohovitosť použijeme kosínus uhla medzi dvomi k-vektorami, ktorý definujeme ako [10] :

$$c_{ik} = \frac{(a_{ik} \cdot b_{ik})}{|a_{ik}||b_{ik}|}$$

Výber : Nebudeme sa zaoberať všetkými bodmi N, ale vyberieme si len zlomok z nich , ktorý označíme M tak že $|M| = |N|\kappa$ a ich počet bude m. Kde κ je užívateľom definovaný parameter. Pre tieto body M vypočítame najlepšiu rohovitosť. Najlepšiu rohovitosť bodu $p_i \in M$ vypočítame ako veľkosť uhla medzi najlepšími ramenami rohu pre bod p_i. Pod pojmom najlepšie ramená bodu p_i budeme rozumieť takú veľkosť $h(i)$

$$c_{im} < c_{im-1} < .. < c_{ih(i)} > c_{ih(i)} > c_{i(h(1)-1)}$$

Inými slovami, ramená takej veľkosti, že rohovitosť ich bodu bude čo najväčšia. Povieme, že bod p_i je rohový, pokiaľ jeho rohovitosť s najlepšími ramenami je lokálne maximum alebo minimum spomedzi najlepších rohovitostí ostatných bodov. Lokálnym maximom budeme rozumieť :

$$\forall j, p_j \in M : |i - j| < h(j)/2 \Rightarrow c_{ih(i)} \leq c_j h(j)$$

Lokálne minimom budeme rozumieť :

$$\forall j, p_j \in M : |i - j| < h(j)/2 \Rightarrow c_{ih(i)} \geq c_j h(j)$$

Parameter je len jeden κ, ktorá špecifikuje zlomok bodov, ktoré sa budú spracúvať. Zvolená je hodnota $\kappa = 0.05$ [12].

Algoritmus Rosenfeld and Weszka RW 75

Algoritmus je veľmi podobný predošlému. Predošlý algoritmus môže viesť k chybným výsledkom, keď sú hrany príliš blízko pri sebe. Modifikovaný algoritmus RW75 popísaný v [11] napráva túto chybu.
Rohovitosť : Priemerný kosínus uhlov medzi k-vektormi definujeme ako :

$$\mathrm{c}_{ik} = \begin{cases} \frac{2}{k+2} \sum_{t=k/2}^{k} c_{it} & \text{ak } k \text{ je párne} \\ \frac{2}{k+3} \sum_{t=(k-1)/2}^{k} c_{it} & \text{ak } k \text{ je nepárne} \end{cases}$$

Výber a **parameter** sú totožné ako v algoritme RJ73.

Z bodov, ktoré dostaneme ako výsledok algortmov RJ73 a RW75 dostaneme výsledné rohové body nasledujúcim spôsobom: Do výsledku pridáme všetky body z algoritmu RW75. Následne pridáme také body z algoritmu RJ73, ktorých indexy sú od indexov bodov RW75 vzdialené aspoň vzdialenosť l. Bod $rj73_i$ (bod, ktorý bol algoritmom RJ73 označený za rohový a v krivke je na mieste i) je pridaný do výsledku ak neexistuje bod $rw73_j$ taký, že $|i - j| < l$. Parameter β je defaultne nastavený na veľkosť 6. Týmto spôsobom zabezpečíme malý počet dominantných bodov a vyhneme sa zgrupovaniu rohových bodov blízko pri sebe.

4.1.2 Detekcia oblúkov na priamke

Románske písmo sa neskladá len z písmen s ostrými prechodmi na rohoch, ale aj z poloblúkov a iných kriviek. Celú krivku si rozdelíme na podkrivky :

$$K_i = \{p_i | p_i \in C_i, r_i > p_i > r_{i+1}\}$$

kde r_i a r_{i+1} sú rohové body. Na každú podkrivku K_i aplikujeme algoritmus na aproximáciu krivky. Keďže už máme zistené rohové body, body vzniknuté touto aproximáciou označíme za body krivky.

Algoritmus Ramer–Douglas–Peucker

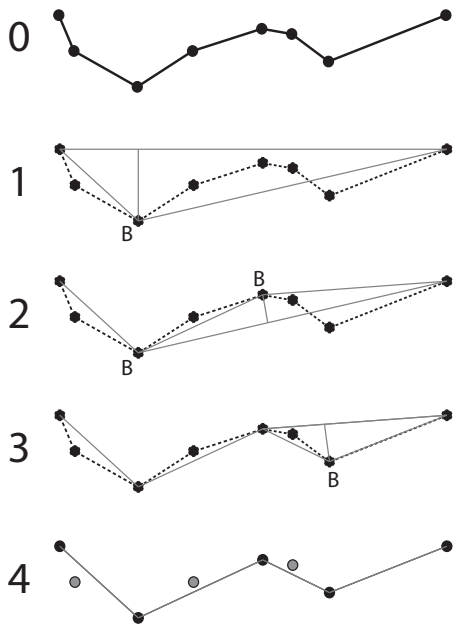
Ako algoritmus na aproximáciu krivky použijeme algoritmus Ramer–Douglas–Peucker [13]. Idea tohto algoritmu je taká, že všetky body pospájame úsečkami a snažíme sa nájsť podobný útvar s menším počtom úsečiek. Podobnosť definujeme ako maximálne možnú vzdialenosť pôvodnej úsečky od novej úsečky.

Algoritmus rekurzívne delí krivku. Začína všetkými bodmi medzi prvým a posledným bodom. Prvý a posledný bod si označíme za body krivky (až na prvý beh, kde prvý a posledný bod už sú rohové body). Následne nájde bod, označíme ho B, ktorý je najďalej od úsečky tvorenej začiatočným a konečným bodom.

- Ak je bod B bližšie ako parameter, tak ukončíme rekurziu, keďže sme vybrali bod, ktorý je najďalej a ani ten nie je dostatočne ďaleko, aby sa porušila podobnosť útvaru bez tohto bodu. Ponecháme teda len začiatočný a konečný bod.

- Ak je bod B vzdialený viac ako parameter, označíme tento bod ako bod krivky a rekurzívne zavoláme algoritmus na podkrivky, ktoré vzniknú z pôvodnej krivky rozdelením bodom B. Takže ho zavoláme na krivku od začiatočného bodu do bodu B a krivku od bodu B do konečného bodu.

Keď je rekurzia kompletná, výstupom sú body zjednodušeného útvaru. Na obrázku 4.2 je názorne ukázané ako algoritmus funguje. V 4.2.1 máme útvar, ktorý budeme aproximovať. Vyberieme bod B, ktorý je najdalej od úsečky ktorá vznikne spojením začiatočného a koncového bodu útvaru. Bod B je ďalej ako parameter, takže ho označíme ako bod krivky a pokračujeme na podkrivkách. V obrázku 4.2.1 máme výsledný útvar. Slabo sivé body sme v priebehu aproximácie vyradili. Čierne body sú body krivky.



Obr. 4.2: Ukážka fungovania algoritmu Ramer–Douglas–Peucker

4.1.3 Komplexný prevod

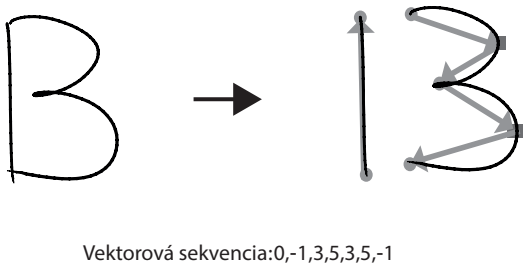
Celý prevod na sekvenciu teda funguje nasledovne. Pre každý ťah vykonáme:

1. Zistíme rohové body použitím algoritmov RJ73 a RW75.
2. Rozdelíme ťah na základe rohových bodov na podkrivky a pre každú podkrivku použijeme algoritmus Ramer–Douglas–Peucker na zistenie bodov krivky.
3. Rohové body a body krivky zjednotíme a získame dominantné body. Z dominantných bodov vytvoríme postupne vektory. Ak vektor vytvoríme len z rohových bodov, označíme ho za "rovný vektor". Ak vektor vytvoríme z čo i len jedného bodu krivky, označíme ho za "krivý vektor".
4. Vektory prevedieme cez Freemenove kódovanie na sekvenciu čísiel.
5. Zlúčime čísla vektorov tak, aby $c_i \neq c_{i+1}$.
6. Pridáme na koniec -1, čo je špeciálne číslo slúžiace na oddelenie ťahov.

Ukážeme si prevod na konkrétnych znakoch: písmene A (obr. 4.3) a písmene B (obr. 4.4):

Obr. 4.3: Prevod písmena A

Napíšeme písmeno A dvomi ťahmi tak, ako vidíme na obrázku 4.3. Na prvom ťahu nájdeme rohový bod. Vytvoríme dva "rovné vektory" a to začiatočný bod - rohový bod a rohový bod - konečný bod. Tieto dva vektory prevedieme Freemanovým kódovaním na sekvenciu 1, 3 a pridáme -1 ako oddelovač ťahov. Druhý ťah nemá žiadny rohový bod ani bod krivky. Vytvoríme s neho jeden vektor začiatočný bod - konečný bod. Tento vektor prevedieme Freemanovým kódovaním na sekvenciu 2 a pridáme -1 ako oddelovač ťahov. Výsledná vektorová sekvencia bude 1, 3, -1, 2, -1.



Obr. 4.4: Prevod písmena B

Napíšeme písmeno B dvomi ťahmi tak, ako vidíme na obrázku 4.4. Prvý ťah je zvislá úsečka vedená zdola hore. Nemá žiadny rohový bod a ani bod krivky. Freemanov kód, ktorý prislúcha jej vektoru je 0 a -1 ako oddeľovač ťahov. Druhý ťah, ktorý sú "bruška" obsahuje jeden rohový bod. Tento bod rozdelí ťah na dve podkrivky, z ktorých každá obsahuje jeden bod krivky (na obrázku 4.4 znázornené šedým štvorcom). Pospájaním všetkých dominantých bodov druhého ťahu do vektorov dostaneme postupne štyri "krivé vektory", ktoré majú Freemanov kód 3,5,3,5 a -1 ako oddeľovač ťahov. Výsledná sekvencia pre B je teda 0, -1, 3, 5, 3, 5, -1.

4.2 Pozičná sekvencia

Prevod kriviek na vektorovú sekvenciu vypovie veľa o tvare písmena. Táto informácia ale úplne neopisuje ich tvar. Príkladom nám môžu byť písmená D a P, ktoré sú vektorovo totožné pri istom spôsobe písania. Riešenie tohto problému je zachytiť informáciu o relatívnej polohe ťahov voči sebe, najlepšie vo forme sekvencie čísiel.

Nebudeme sa zaoberať pozíciou každého bodu krivky, pretože o jej tvare nám hovorí vektorová sekvencia. Budeme sa zaoberať len tromi druhmi bodov :

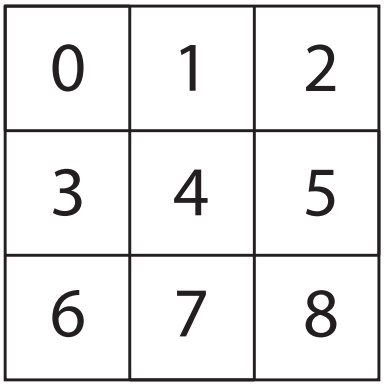
1. Začiatočné body - body, kde začneme písať ťah.
2. Konečné body - body, kde skončíme písanie ťahu.
3. Prienikové body - body, kde sa jeden ťah kríži sám so sebou.

Tieto body sú pre nás dostatočné. Vypovedajú, kde sme začali a skončili písať, a kde sme ťah prekrížili.

4.2.1 Prevod ťahov na pozičnú sekvenciu

Prvým krokom je ohraničenie aktuálnych ťahov do obdĺžnika. Algoritmus si nemusíme implementovať sami, je prítomný priamo v operačnom systéme pre Tablet PC. Jeho implementácia nie je ťažká. Stačí nájsť 4 body, z toho 2 s najmenšou/najväčšou súradnicou Y, 2 s najmenšou/najväčšou súradnicou Y a opísať ich obdĺžnik.

Druhým krokom je rozdelenie obdĺžnika na 9 rovnakých segmentov. Rozdelenie na 9 segmentov (3x3) je vybrané preto, že pri písaní budeme rozlišovať 3 polohy: hore, stred, dole a analogicky prevrátené na horizontálnu os.



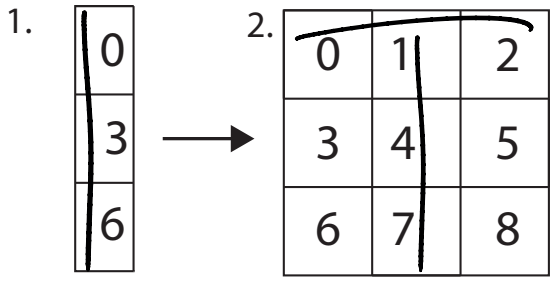
Obr. 4.5: Rozdelenie obdĺžnika na segmenty

Sekvencia bodov je potom určená na základe vzdialenosti od ľavej a spodnej strany obdĺžnika.

Sekvenciu by sme mohli zapísať ako 2 čísla, X-ový a Y-ový index segmentu, ale pre zachovanie rovnakého prístupu ako pri vektorovej sekvencii ich prevedieme na čísla [0..8] (obr. 4.3).

Pri zadávaní znaku, ktorý pozostáva z viacerých ťahov, spracúvame ťahy sekvenčne. Najprv spracujeme prvý ťah na sekvenciu. Keď zadáme ďalší ťah, ohraničíme všetky doterajšie ťahy tak, aby všetky boli ohraničené jedným obdĺžnikom. Keďže sme zmenili obdĺžnik, podľa ktorého sme počítali pozičnú sekvenciu, musíme prepočítať všetky ťahy.

Máme daný minimálny rozmer segmentu. Na šírku (alebo výšku) musí jeden segment pokryť úsečku sklopenú pod uhlom, ktorý je definovaný ako odchýlka od osi pri Freemanovom kódovaní. Týmto si zabezpečíme to, že trochu nahnutá úsečka, ktorú plánujeme písať ako horizontálnu, bola takto rozoznaná aj pozične.



Obr. 4.6: Pozičná sekvencia pre T

Na obrázku 4.6 vidíme ako prebieha výpočet pozičnej sekvencie pre písmeno T. Pri prvom ťahu je pozičná sekvencia $0, 6, -1$. Pri pridaní druhého ťahu je pozičná sekvencia $1, 7, -1, 0, 2, -1$.

4.2.2 Alternatívne výsledky pozičnej sekvencie

Prevod pozície na číselnú sekvenciu je veľmi náchylný na chyby, keď sa bod nachádza blízko hranice segmentu. Toto vedie k nesprávnej pozičnej sekvencii a následne k nesprávne rozpoznanému znaku.

Na zabránenie tejto chyby si zavedieme ďalšiu podpornú sekvenciu čísiel nazvanú maska. Masku sekvencie nebudeme používať na opis znaku tak ako pozičnú alebo vektorovú sekvenciu, ale len ako prevenciu proti chybám. Masku sekvencie nebudeme ukladať do žiadnej dátovej štruktúry slúžiacej na uloženie naučených znakov. Každý segment rozdelíme na ďalších 9 podsegmenov očíslovaných rovnako ako segment, -1 je oddeľovač ťahov. Maska nám slúži na zaznačenie, v ktorej časti segmentu sa bod nachádza. Na základe tejto pozície vieme vytvoriť alternatívny výsledok.

Podľa toho v akej časti segmentu sa nachádza, zmeníme pôvodné zaradenie do segmentu na segmenty, ktoré sú najbližšie. Napríklad pre pôvodný segment 4 a masku 0 budeme mať tri alternatívne výsledky a to $0, 3, 1$ a pôvodný výsledok 4 . Alternatívny výsledok celej analýzy bude kombinácia všetkých možných alternatívnych výsledkov každej sekvencie. Týmto sa nám zvýši rapídne počet výsledkov.

Alternatívne výsledky budeme používať vtedy, keď nenájdeme pôvodnú poziční sekvenciu.

4.3 Kompenzácia ľudských chyb pri písaní

Pri testovaní prevodu kriviek na dominantné body sa vyskytovali chyby, ktoré neboli spôsobené nesprávnou implementáciou, ale vstupmi, ktoré obsahovali chyby. Tieto chyby vyplývali zo spôsobu písania na tablete. Najzávažnejšie chyby boli deformovaná rovná čiara a úvodné háčiky, ktoré bolo nutné kompenzovať.

Kompenzácia rovnej čiary

Písanie značkou na rovnaké miesto, s rukou položenou na tom istom mieste nesie so sebou jeden problém - písanie horizontálnych rovných čiar. Oveľa prirodzenejší pohyb je písanie oblúkov so stredom okolo bodu, v ktorom je ruka položená na podložke. V takomto prípade len otočíme rukou. Naopak, keď píšeme rovnú čiaru, musíme zároveň otáčať rukou okolo bodu, kde je ruka položená na podložke, a taktiež upravovať vzdialenosť pera od tohto bodu natiahnutím prstov, ktoré držia pero alebo ruku z podložky zdvihnúť.

Problém je zistiť, kedy užívateľ plánoval písať oblúk, a kedy nie. Príkladom, kedy ide o nejasný problém je rozdiel medzi Z a 2, kde pri istom obľúbenom spôsobe písania je rozdiel len v ťahu hornej úsečky.

Obr. 4.7: Rozdiel medzi 2 a Z

Túto chybu budeme kompenzovať nasledujúcim spôsobom. Pre každé dva rohové body, označíme si ich r_1 a r_2 , si zoberieme všetky body kriviek $c_1..c_n$, ktoré sú medzi nimi. Postupne si vypočítame vzdialenosťi d_i všetkých bodov c_i

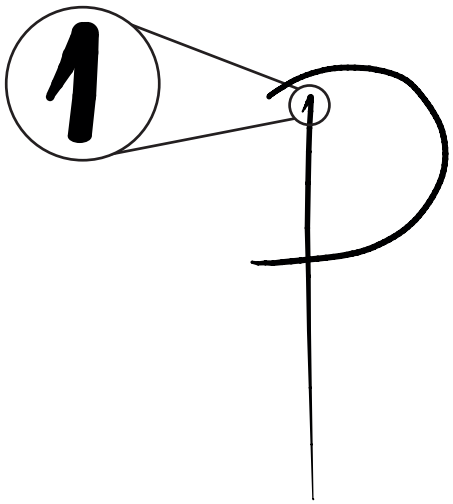
$$l = vzdialenos(r_1; c_i) + vzdialenos(r_2; c_i);$$

Ak táto vzdialenosť je menššia ako parameter θ , tak tento rohový bod odstánime. Týmto spôsobom si určíme maximálnu elipsu, ktorú vykreslí ruka pri prirodzenom pohybe. Body, ktoré sú nad touto elipsou boli teda napísané zámerne. Bodý, ktoré boli pod touto elipsou sú napísané neúmyselne. Ako parameter maximálnej vzdialenosti θ si zvolíme t-násobok vzdialenosti medzi r_1 a r_2. Defaultne t = 1,18 .

Kompenzácia zle začatej krivky

Závažnejším problémom je, že si ruka spoločne s perom pri pohybe zachováva určitú hybnosť. Keď potom priložíme stylus na povrch tabletu zachováva si pôvodný smer na malú vzdialenosť, ktorá je ale pri presnosti digitalizéra zaznamenaná (obr. 4.8).

Toto nie je problém, ak píšeme ťah rovnakým smerom, akým sme predtým presúvali ruku. Obyčajne sa to ale nedeje. Spravidla je to práve opačný smer, či už pri začiatku písania alebo pri konci. Keď zdvihneme stylus neskoro vznikajú na koncoch krivky „fajky“. „Fajky“ sú útvary, kde sa na krátkej vzdialenosti zmení zásadnejšie smer ťahu.



Obr. 4.8: Chyba pri písaní P

Tento problém budeme kompenzovať tak, že odstránime každý segment kratší ako je určitý zlomok z priemernej dĺžky segmentov v rámci celého útvaru.

4.4 Spracovanie dominantných bodov na znak

Pre spracovanie dominantných bodov na znak nám treba ešte vyriešiť ukladanie naučených znakov, ako ich budeme nasledovne vyhľadávať, a ako budeme riešiť nejednoznačnosť. Pre každý znak budeme môcť nahrať viacero spôsobov jeho písania. K jednotlivým sekvenciám pre ich rozdielne vlastnosti musíme pristupovať odlišne.

4.4.1 Vektorová sekvencia

Pri vektorovej sekvencii je veľmi dôležitý kontext. S pridaním ďalšieho ťahu sa sekvencia, ktorá je dovtedy vypočítaná nemení (nenachádzame sa v koreňovom uzle) len v nej pokračujeme. Na základe tohto faktu na uloženie a vyhľadávanie použijeme strom ako dátovú štruktúru, v ktorej budeme ukladať vektorové sekvencie. Každý uzol v strome reprezentuje istú konečnú sekvenciu. Každá sekvencia začína v koreni, ktorý reprezentuje nultú pozíciu sekvencie. Nultú pozíciu

sekvencie budeme chápať ako začiatok písania nového znaku, budeme vyhladávať v strome od koreňa. Sekvenciu $s_0..,s_{n-1},s_n$ reprezentuje uzol na n-tej hladine stromu, ktorý má hodnotu s_n a jeho predok reprezentuje sekvenciu $s_0..s_{n-1}$
Každý uzol tohto stromu bude obsahovať:

- Zoznam znakov, ktorých sekvencie končia v danom uzle. Tieto znaky označíme ako koncové znaky.
- Zoznam uzlov, ktoré sú potomkami daného uzla.
- Hodnotu posledného prvku sekvencie, ktorý uzol reprezentuje.
- Zoznam všetkých písmen, ktorých vektorová sekvencia je rovnaká do pozície, akú má úroveň tento uzol, ale v danom uzle nekončia. Zároveň toto písmeno je tvorené rovnakým počtom ťahov ako sekvencia, ktorú reprezentuje uzol. Tieto znaky označíme ako nekoncové znaky.

Pri vyhľadávaní napíšeme znak. Písanie znaku začína prvým ťahom, ktorý sa prevedie na vektorovú sekvenciu. Na základe tejto vektorovej sekvencie vyberieme uzol, ktorý ju reprezentuje a vrátime z tohto uzla zoznam koncových znakov. Ak nevieme nájsť podľa tejto vektorovej sekvencie uzol, tak si vyberáme uzol, v ktorom vyhľadávanie zlyhalo, teda taký uzol, ktorý reprezentuje sekvenciu $s_0..s_{n-u}$, ale už neexistuje uzol, ktorý reprezentuje sekvenciu $s_0..s_{(n-u)+1}$. Z tohto uzla vrátime zoznam nekoncových znakov. Pri pridaní ďalšieho ťahu pokračujeme vyhľadávaním sekvencie od posledného nájdeného uzla.

4.4.2 Pozičná sekvencia

Pri vektorovej sekvencii nie je dôležitý kontext. Pridaním ďalšieho ťahu sa môže a spravidla sa mení celá pozičná sekvencia. Nemôžeme si teda prepočítať pri prvom ťahu časť vyhľadania. Ako dátovú štruktúru budeme používať slovník (dictionary). Je to dátová štruktúra, ktorá podľa kľúča vráti uložený objekt. Ako kľuč si zvolíme zoznam celých čísiel (sekvenciu) a ako objekt si zvolíme zoznam znakov. Znak označíme ako koncový znak vtedy, keď bude pozičná sekvencia konečná, teda pozostávajúca zo všetkých ťahov ktorými píšeme znak. Znak označíme ako nekoncový znak, kde jeho pozičná sekvencia pozostáva len z niektorých ťahov. Pri vyhľadávaní, keď napíšeme prvý ťah daného znaku, sa vyhľadá v slovníku podľa pozičnej sekvencie zoznam znakov.

4.4.3 Spracovanie sekvencií

Zjednodušili sme napísané krivky (znak) na dve sekvencie čísiel. Jedna zaznamenáva smer písania a druhá zaznamenáva pozície ťahov pri písaní. Posledným krokom je na základe týchto sekvencií zistiť, aký znak bol napísaný. Budeme postupne zmenšovať množinu potencionálnych znakov. Keď napíšeme prvý ťah, vypočítame pozičnú sekvenciu. Podľa kľúča vo forme pozičnej sekvencie vyhľadáme v slovníku množinu znakov, nazveme ju pozičné znaky. Ak je táto množina prázdna, použijeme masku pozičnej sekvencie a vygenerujeme si všetky možné pozičné sekvencie, ku ktorým nájdeme pozičné znaky. Vektorové znaky získame

obdobne výpočtom vektorovej sekvencie a nájdením vektorových znakov v strome.

Spravíme si prienik týchto dvoch množín. Ak je výsledkom jeden znak, tento znak prehlásime za výsledný. V opačnom prípade sa pozrieme najprv na pozičné znaky a následne na vektorové znaky, či sa v nich nenachádza pravé jeden koncový znak. Ak áno, prehlásime ho za výsledný. Ak je počet výsledných znakov dva, spracujeme to podľa kapitoly Nejednoznačnosť dvojíc znakov (4.4.4). Ak ich je viacej spracujeme ich podľa kapitoly Nejednoznačnosť obecne (4.4.5). Ak je prienik prázdny nevieme určiť aký znak sme napísali.

4.4.4 Nejednoznačnosť dvojíc znakov

Niektoré dvojice znakov nie je možné od seba rozlíšiť na základe vektorovej sekvencie, ani pozičnej sekvencie, majú ju totiž totožnú. Tieto dvojice sa pokúsime rozšíriť na základe iných vlastností a to napríklad podľa počtu bodov, výskytu rohových bodov alebo bodov kriviek. Problematické sú dvojice znakov :

- U a V, kde sa rozhodneme pre U, ak je aspoň jeden domintný bod znaku bod krivky.
- 0 a O, kde sa rozhodneme pre 0, ak je pomer výška/šírka väčší ako užívateľom definovaný pomer.
- B a D, kde sa rozhodneme pre B, ak je znak písaný jedným ťahom a sú aspoň dva domintné bod znaku bodmi krivky. Ak je znak písaný dvomi ťahmi a je aspoň jeden domintný bod znaku bod krivky.

4.4.5 Nejednoznačnosť obecne

Stáva sa, že zadáme znak, ktorý sa presne nezhoduje s tým, ktorý sme si nahrali. V tomto prípade sa ani vo vektorových znakoch, ani v pozičných znakoch nevyskytne jedna úplná zhoda. Máme teda množinu potencionálnych znakov, ktorú si vytvoríme zjednotením vektorových znakov s pozičnými znakmi. Chceme teraz vedieť, ktorý potencionálny znak sa najviac podobá na nami vložený znak.

Podobnosť si zadefinujeme ako najmenšiu hodnotu Levenshtein-ovej editačnej vzdialenosti [**?**] medzi vektorovou sekvenciou neznámeho znaku a každou nahranou vektorovou sekvenciou pre každý potencionálny znak. Teda pre každý potencionálny znak Z si zoberieme všetky vektorové sekvencie, ktoré sme si nahrali. Pre každú takúto vektorovú sekvenciu si vypočítame Levenshteinove vzdialenosti vzhľadom na vektorovú sekvenciu neznámeho znaku. Najmenšia takáto vzdialenosť je hodnota podobnosti znaku Z k neznámemu znaku.

Znak s najmenšou podobnosťou, ktorý bude práve jeden, a ktorého podobnosť bude menšia ako maximálna prípustná podobnosť označíme za vybratý. V prípade, že znakov s najmenšou podobnosťou bude viac ako jeden, alebo podobnost najpodobnejšieho znaku bude väčšia ako maximálna prípustná podobnosť, nevrátime žiadny koncový znak. V taktomto prípade ďalej čakáme na ďalšie ťahy, alebo na začanie písania nového znaku.

Levenstainova editačná vzdialenosť

Editačná vzdialenosť je spôsob merania rozdielu medzi dvomi slovami (sekvenciami znakov). Levenstainova editačná vzdialenosť medzi dvomi sekvenciami je definovaná ako minimálny počet zmien, ktoré treba vykonať nad jednou sekvenciou, aby sme dostali sekvenciu druhú. Sú povolené len nasledujúce operácie : vloženie nového znaku, vymazanie znaku alebo substitúcia znaku.

Výpočet Levenstainovej vzdialenosti vykonáme pomocou techniky dynamického programovania. Tá pracuje nad maticou o rozmere prvej sekvencie m krát rozmer druhej sekvencie n. Prvý riadok a prvý stĺpec vyplníme postupne číslami $1..m, 1..n$. Ďalšie políčka matice sa plnia následovne.Postupne zväčšujeme index prvého reťazca i a druhého reťazca j. Ak sa znaky na pozíciách i a j rovnajú, nevykonávame žiadnu transformáciu a len skopírujeme, hodnotu na pozícii $i-1$ a $j-1$. Udáva nám to, potrebný počet transformácií aplikovaných od začiatku do daného stavu. Ak sa nerovnajú, pozrieme sa o stĺpec vyššie a vyberáme políčko s najmenším číslom, ktoré reprezentujú operácie vloženie, zmazanie a substitúciu. Výsledkom je hodnota tabulky na pozícii $[m, n]$

4.4.6 Nerozhodnosť znakov

Pri zadávaní znakov, ktoré pozostávajú z viacerých ťahov sa môže stať, že jeden znak je podznakom iného, totiž že jeden jednoznačne určený znak sa dá rozšíriť ďalším ťahom na iný jednoznačne určený znak. Takýmito znakmi sa často stavajú P a R alebo E a F pri istom spôsobe písania. Tento prípad budeme riešiť následovne :

- Pri zadávaní poziˇcnej sekvencie, keď budeme pridávať nový znak sa vždy pozrieme, či koncová sekvencia nového znaku nie je nekoncová sekvencia iného znaku. Ak áno, tak nový znak zmeníme z konečného na konečne nerozhodný.
- Pri vyhľadávaní, ak nájdeme konečný nerozhodný znak, tak sa tento znak uloží a spustí sa časovač. V oblasti, kde sa zadávajú znaky sa zobrazí notifikácia spustenia časovača. Východzia hodnota časovača je 300 ms.
- Ak tento čas uplynie, uložený znak sa označí za vybraný.
- Ak budeme pokračovať v písaní, zruší sa časovač. Ak napíšeme znak, ktorý bude rozoznaný, tento nový znak sa označí za vybraný. Ak ale nový znak nebude rozoznaný, uložený znak označíme za vybraný a posledný ťah bude prvým ťahom nového znaku.

Tento postup je jediné použitie akcie, ktorá je oddialená v čase. Oddialením v čase získame priestor na rozhodnutie, aký znak chceme napísať. Ak ale nechceme prepisovať znak, tak len jednoducho začneme písať ako na papier vedľa pôvodného znaku. Takto definovaný znak sa nerozozná a rozdelí sa na znak, ktorý je uložený a na prvý ťah ďalšieho znaku.

5. Softwarová implementácia pre systém Windows

Vysoko odporúčame používať program výhradne na počítači Tablet PC alebo aspoň na počítači vybaveným externým digitalizérom najlepšie značky Wacom. Pri použití myši nie je zaručená úplná funkčnosť a celkový dojem z aplikácie.

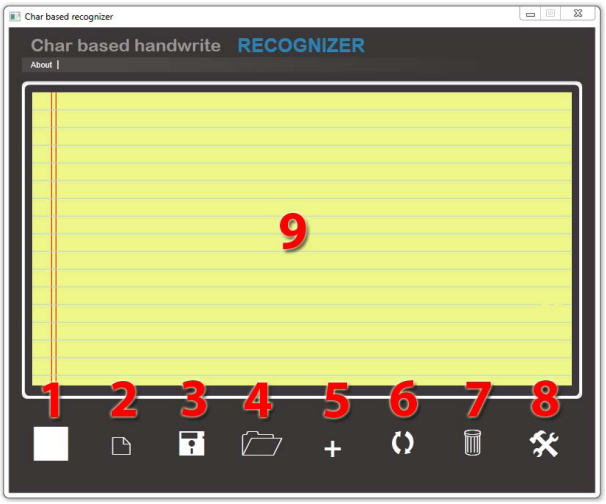
Aplikácia nepodporuje diakritiku, vzhľadom na zameranie to nebudeme považovať za nedostatok.

5.1 Užívateľská príručka

5.1.1 Inštalácia

Program sa do systému neinštaluje, stačí rozbaliť archív. Ten obsahuje samotný program, ako aj podporné knižnice. Na spustenie je nutné mať nainštalovaný .NET framework vo verzii 4.0, ktorá je dostupná na CD.

5.1.2 Používanie



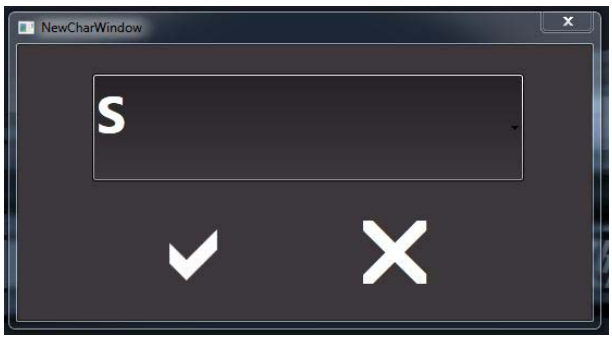
Obr. 5.1: Hlavné okno programu

Program sa pri spustení používateľa spýta na profil písania písmen, základný profil je dostupný na CD. Po zapnutí je nutné v nastavení vybrať aplikáciu, do ktorej budeme vkladať text. Viac v kapitole nastavenia (5.1.3).

Ak si chceme nahrať úplne nový profil, vyberieme ikonku vytvorenia nového profilu (obr. 5.1 - 2). Otvorí sa nám okno uloženia do ktorého napíšeme meno nového profilu a klikneme na OK. Iný profil naháme kliknutím na ikonu nahrania profilu (obr. 5.1 - 4). Otvorí sa nám okno výberu súboru v ktorom vyberieme súbor s

príponou .crup(C.har R.ecognizer U.ser P.rofile). Na ukladanie profilu slúži ikonka uloženia (obr. 5.1 - 3).

Keď chceme nahrať spôsob písania nového znak, stlačíme ikonku pridania (obr. 5.1 - 5). Ikona sčervenie, a tým sa dostaneme do módu pridávania. tomto móde vložené ťahy budú len analyzované, nebudú sa na ich základe vyhľadávať znaky. Keď sme spokojní s tvarom novonapísaného znaku, opäť klikneme na červenú ikonu vloženia. Objaví sa nám okno (obr. 5.2), v ktorom si vyberieme aký znak chceme nahrať a potvrdíme fajkou.



Obr. 5.2: Výber znaku

Keď je nesprávne rozoznaný znak,alebo sme začali písať znak nesprávne, môžeme rozoznanie zmazať ikonou mazania (obr. 5.1 - 1). Nesprávne rozoznanie môžeme opraviť. Opravíme ho stlačením ikony editácie (obr. 5.1 - 6). Ikona zozelenie a prepneme sa do módu editácie. V ploche pre písanie (obr. 5.1 - 9) sa nám objavia ťahy, ktoré boli zle analyzované. V tomto momente môžeme dopísať ďalšie ťahy. Na uloženie klikneme na zelenú ikonu editácie. Objaví sa nám okno výberu znaku . Potvrdíme fajkou.

Keď nie sme spokojní s natrénovaním aplikácie na konkrétny znak, môžeme tento znak z profilu zmazať. Na vykonanie tohto kroku klikneme na ikonu zmazania (obr. 5.1 - 7). Objaví sa nám okno výberu znaku (obr. 5.2). Vyberieme si znak, ktorý chceme zmazať (zmažú sa všetky formy jeho písania). Potvrdíme fajkou.

5.1.3 Nastavenie

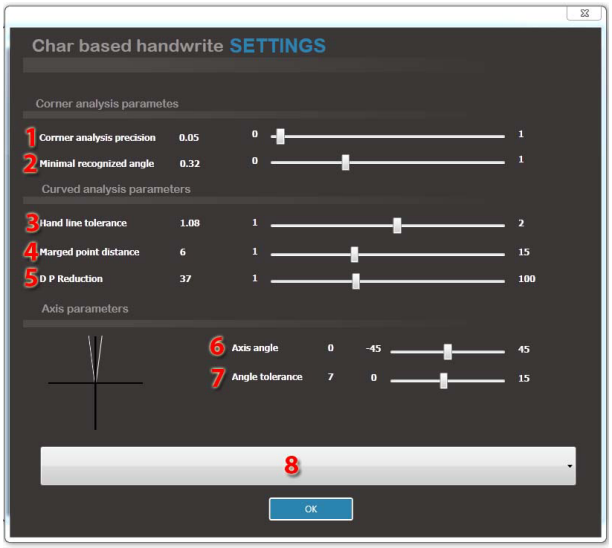
Ikonka nastavenia (obr. 5.1 - 7) slúži na vstup do nastavenií (obr. 5.3). Pri každom parametri, keď poklepeme dvakrát na názov parametru, vráti sa jeho hodnota na východziu hodnotu.

Program, do ktorého budeme zadávať text vyberieme kliknutím na rozbaľovaciu ponuku (obr. 5.3 – 9).

Nastavenie presnosti analýzy rozoznávania rohov nastavíme v parametroch, ktoré sú združené pod nadpisom Corner analysis parametes. Parameter κ v algoritmoch RJ73 a RW75 sa nastavuje posuvným voličom (obr. 5.3 – 1). Keď chceme nastaviť, aby aj menšie rohy boli rozoznané, zmeníme parameter minimálneho rozpoznateľného uhlu, ktorý prehlásime za roh (obr. 5.3 – 2).

Nastavenie presnosti analýzy kriviek sa nastavuje v parametroch, ktoré sú združené pod nadpisom Curved analysis parameters. Veľkosť kompenzácie rukou neplánovaného oblúku nastavíme parametrom (obr. 5.3 – 3). Číslo vedľa voliča nám udáva koľkokrát môže byť dĺžka medzi stredovým bodom a krajnými bodmi väčšia ako dĺžka medzi krajnými bodmi. Parametre analýzi nastavíme parametrami (5.3 – 4) pre minimálnu vzdialenosť bodov krivky od seba. Jemnosť aproximácie krivky algoritmom Ramer–Douglas–Peucker nastávíme voličom (obr. 5.3 – 5). Jedná sa o percentuálny podiel maximálnej vzdialenosti použitek pri podobnosti v algoritme Ramer–Douglas–Peucker vzhľadom na vzdialenosť krajných bodov.

Sklon písania môžeme nastaviť sklonením osi pre prevod na Freemanovu sekvenciu parametrom (obr. 5.3 – 6). Vôľu tolerancie, čo ešte budeme považovať za ťah v jednom z hlavných smerov nastavíme parametrom (obr. 5.3 – 7).



Obr. 5.3: Nastavenia programu

5.2 Programátorská príručka

Riešenie je implementované na platforme Windows, konkrétne nad operačným systémom Windows 8. Ako jazyk implementácie budeme používať C#, konkrétnejšie verziu .NET 4.0.

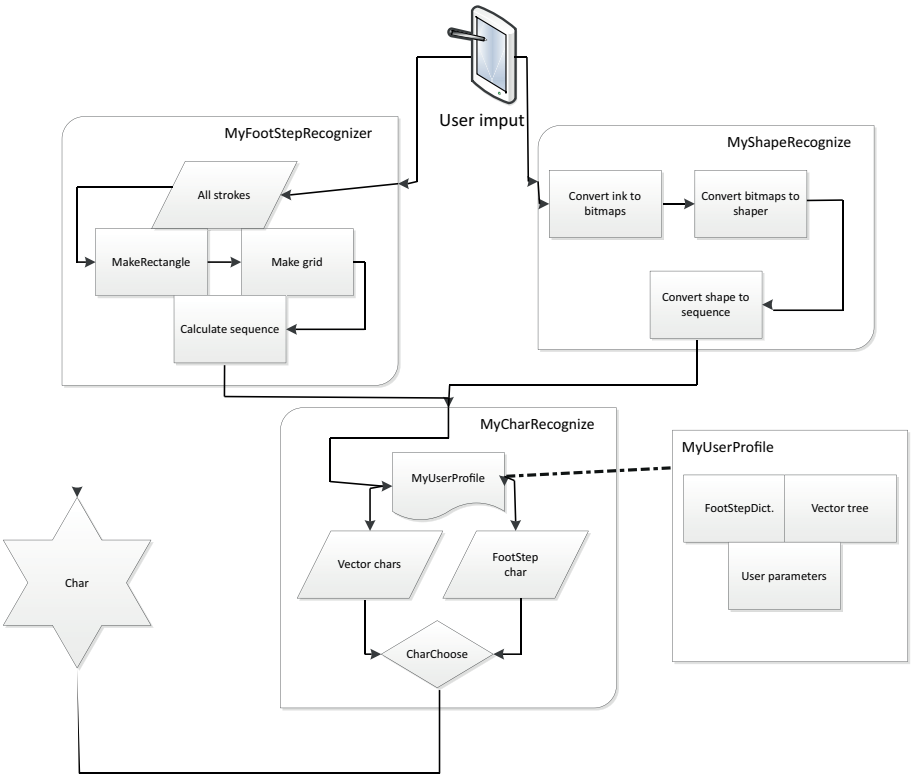
V C# je kontroverzný prístup k digitálnemu atramentu v priebehu verzií.
Pri predstavení platformy Tablet PC ešte na operačnom systéme Windows XP bol predstavený framework pre Tablet PC, ktorý obsahoval pokročilé funkcie ako nájdenie prienikov, bazierových bodov a podobne. Tento framework nie je ale kompatibilný s novšími verziami .NETu ani grafickými objektami, ktoré sa používajú na zaznamenávanie digitálneho atramentu vo Windows Presentation Foundation. Windows Presentation Foundation(WPF) je podmnožina .NETu vytvárajúca bohaté grafické rozhranie. Dátové typy, ktoré uschovávajú a obsluhujú digitálny atrament (Ink) nie sú kompatibilné, a nie je možné previesť Micro-

soft.Ink na System.Windows.Ink a opačne.

S príchodom verzie .NET 4.0 zmizli celé knižnice spracúvajúce digitálny atrament, ktoré boli dostupné v predchádzajúcej verzii 3.5, najmä jeho analýzu a prevod na znaky, ale aj rôzne analýzy ťahov. Situácia je lepšia pri verzii 4.5, ktorá je v čase písania tejto práce vo verzií beta a niektoré pokročilé funkcie fungujú len na operačnom systéme Windows 8, čo by spôsobovalo nekompatibilitu so staršími systémami. Cielom softwarovej implementácie bolo vyvinúť software fungujúci na majoritnej verzii operačného systému.
Pre tieto nešťastné vlastnosti rôznych verzii .NETu je celá práca čo najmenej závislá na externých funkciách spracujúcich digitálny atrament(Ink). Pre túto nezávislosť sme si nadefinovali aj najprimitívnejšie dátové objekty, ako bod alebo vektor sami. Jediným miestom, ktoré je závislé na funkciách digitálneho atramentu je prevod krivky z digitalizéru na zoznam našich bodov (MyPoint).
V nasledujúcich kapitolách sa pozrieme rámcovo na vnútornú organizáciu implementácie. Podrobnejšia programátorská príručka je dostupná na CD prílohe.

5.2.1 Vnútorná organizácia programu

Celý program má tzv. sendvičovú konštrukciu, kde máme nad sebou rôzne vrstvy programu. Každá spracúva vstupné údaje a posunie ich nižšej vrstve. Najlepšie jeto vidieť na obrázku (obr. 5.1). Názvy začínajú predponou My- pre zachovanie istej politiky názvov, ktorá vznikla pri implementácií jednoduchých dátových typov ako vektor, ktorý je nazvaný MyVector.



Obr. 5.4: Schéma programu

MyShapeRecognizer

Každý ťah je predaný triede MyShapeRecognizer. Ťah je transformovaný postupne. Najprv prevedieme ťah na zoznam bitmáp použitím algoritmov na získanie dominantných bodov. Každá bitmapa je pole boolov o rovnakej dĺžke, ako je počet bodov vstupného ťahu. Ak je v i-tej bitmape na j -tom mieste hodnota true, hovorí nám to, že j -tý bod má vlastnosť i. Máme tri bitmapy. Prvá označuje začiatočný a konečný bod. Druhá označuje rohové body. Tretia označuje body krivky.

Zoznam bitmáp je transformovaný na sekvenciu. Zo zoznamu bitmáp je vytvorený objekt MyShape, ktorý reprezentuje rozanalyzovaný tvar znaku. V rámci triedy MyShape prevedieme vybrané body na segmenty (MyLine), kde ich rozlíšime na krivky (CurvedLine) a úsečky (StraightLine). Nasledovne priradíme k týmto segmentom Freemanove kódy a zjednodušíme ich, čím získame ako výsledok vektorovú sekvenciu.

MyFootStepRecognizer

Každý ťah je predaný triede MyFootStepRecognizer, kde je pridaný na haldu. Vložený ťah je ohraničený obdĺžnikom, ktorý pokrýva nový ťah tak, ako aj všetky ťahy uložené na halde. Tento obdĺžnik je rozdelený a potom je každý ťah z haldy podľa tohto rozdelenia prevedený na pozičnú sekvenciu. V tomto kroku je vypočítaná aj maska sekvencie. V prípade, ak MyCharRecognizer nenájde žiadny pozičný znak, tak na základe masky a sekvencie vráti všetky alternatívy.

MyCharRecognizer

Vektorovú sekvenciu predáme do nižšej úrovne MyCharRegocnizer, kde pretransformujeme sekvencie na výsledný znak. Užívateľské dáta, v ktorých budeme vyhľadávať sekvencie sú uložené v serializovateľnej triede MyUserProfile, ktorá v sebe obsahuje slovník na pozičné sekvencie, tak aj strom na vektorové sekvencie a užívateľom nastavené parametre (MyParams). Trieda FootstepRecognizer nám poskytne pozičnú sekvenciu. Ak k nej nenájdeme žiadny pozičný znak, trieda FootstepRecognizer nám poskytne všetky alternatívne pozičné sekvencie. Alternatívne pozičné sekvencie postupne vyhľadáme v pozičnom slovníku a ich výsledné znaky zjednotíme do pozičných znakov. Následne si vo vektorom strome vyhľadáme vektorové znaky podľa vektorovej sekvecie. Na základe týchto dvoch zoznamov znakov potom vyhodnotíme konečný znak. Rozhodovanie medzi znakmi má na starosti trieda ActionArbiter. Pri nájdení konečného znaku vyvoláme Event o nájdení znaku, ktorý je spracovávaný postupne až do triedy MyShapeRecognizer. V priebehu tohto spracovávania je emulované stlačenie klávesnice pomocou externej knižnice ImputSimulator. Zároveň je zobrazený výsledný znak na obrazovke, na ploche pre zadávanie znakov a sú zmazané všetkých čiastkové výpočty, ako napríklad pozícia vo vektorovom strome, ktoré si predpočítame na možný ďalší ťah.

5.2.2 Výkon programu

Výsledkom implementácie má byť program, ktorý spracúva a analyzuje užívateľom zadané znaky v reálnom čase. Nemôže sa nám teda stať, že užívateľ po napísaní znaku bude na jeho spracovanie čakať príliš dlho. Pre túto vlastnosť musí byť program optimalizovaný na čo najväčšiu rýchlosť.V priebehu programu je použitá veľká paralelizácia výpočtov, ktoré môžu byť paralelizované. Na manažment vlákien použijeme v .NETe integrovaný Threadpool.

Paralelizované sú výpočty rohových bodov do dvoch vlákien pre algoritmus RJ73 a RW75. Algoritmus RW75 je ďalej paralelizovaný pre každý bod samostatným vláknom. Pre algoritmus RJ73 je implementovaný aj paralelný spôsob výpočtu, ale pre relatívne vysokú réžiu prepínania vlákien a malú výpočtovú zložitosť na jeden bod nie je použitá.

Paralelizácia výpočtu bodov krivky algoritmom Ramer–Douglas–Peucker je implementovaná nad každým intervalom medzi rohovými bodmi do samostatného vlákna.

Pri testoch programu je najnáročnejšie na výkon grafické prostredie a najmä efekt "fishpanel". Samotný program zaberá maximálne 17 percent výkonu na testovacom počítači, aj to len krátkodobo. Priemerný počet naalokovaných vlákien na Threadpoole je 25.

5.2.3 Grafický návrh programu

Prívetivý dizajn samotného programu je častokrát dôležitejší ako jeho funkčnosť. Graficky je program ladený do farieb a grafického dizajnu operačného systému Windows 8. Boli použité tmavé farby, bledý kontrastný text a celkovo metro dizajn.

Užívateľ píše na plochu (obr. 5.1 - 9), ktorá má vzhľad bežného listu recyklovaného papiera. Pre lepšie zarovnanie a väčšiu užívateľskú prívetivosť sú na ňom naznačené riadky. Ovládanie programu bolo prispôsobené primárne na ovládanie perom. Pre toto prispôsobenie sú všetky ovládacie prvky dostatočne veľké. Špeciálne sú aj samotné hlavné ikony (fishpanel). Ikona, nad ktorou sa vznáša pero sa zväčší, ostatné ikony v rade sa zmenšia. Tento efekt umožňuje lepšie vidieť obsah ikony a uľahčuje jej zakliknutie.

6. Porovnanie metód zadávania textu

V nasledujúcej kapitole porovnáme naše riešenie s metódami zadávania textu bez hardwarovej klávesnice, ktoré sú štandartne dostupné na operačnom systéme Windows 8. Najprv si predstavíme v porovnávané metódy a následne metodiku testovania. Na záver sa pozrieme na výsledky a postrehy z testovania. Porovnávať budeme :

- Objektívne vlastnosti - rýchlosť písania, presnosť písania
- Subjektívne vlastnosti - jednoduchosť a rýchlosť používania, efektívnosť v simulovanom každodennom použití, celkový dojem.

Ako zariadenie na testovanie budeme používať tablet HP 2710p v nasledujúcej konfigurácií : 12,1" displej, procesor Intel Centrino s frekvenciou 1.2GHz, 3Gb RAM, 4200 otáčkový 80GB pevný disk s operačným systémom Windows 8. Digitalizér je od spoločností Wacom zaznamenávajúci len polohu a dotyky stylusu .

Pred samotným testovaním umožníme testovaciemu subjektu istý čas na oboznámenie sa s ovládaním tabletu perom a dáme mu priestor vyskúšať si metódy zadávania textu mimo samotného testovania. Tento čas bol pri každom subjekte aspoň 10 minút. Subjekty na testovanie boli vybrané bez ohľadu na pohlavie, vek a počítačovú gramotnosť.

6.1 Predstavenie testovaných metód

Prvá testovaná metóda je virtuálna klávesnica. O nej sa dozvieme viac v kapitole 3.1. Dostupné jazyky pre virtuálnu klávesnicu pri testovaní boli slovenčina, angličtina a čeština.

Druhou testovanou metódou bol Tablet PC Input Panel. Táto metóda vstupu využíva vstup vo forme prirodzeného rukopisu alebo tlačeného písaného písma. Metóda využíva jazykový model. Dostupné jazyky sú angličtina a čeština. Slovenčina nie je podporovaná.

Treťou testovanou metódou bol program, ktorý sme implementovali. Program mal dopredu nahranú základnú sadu písmen, ktorú si ale mohol užívateľ mimo samotného testovania prispôsobiť jeho štýlu písania.

Štvrtou metódou bola hardwarová klávesnica. V tomto prípade mohol testovací subjekt ovládať tablet stylusom ale aj vstavaným touchpointom. Touchpoint je malý výstupok citlivý na dotyk, ktorý sa nachádza na klávesnici medzi klávesami G, H, B ktorým sa ovláda pohyb kurzora.

6.2 Objektívne metódy testovania

Testovanie objektívnych vlastností budeme merať pri prepisovaní textu z angličtiny a z rodného jazyka testovaného subjektu o dĺžke približne 200 znakov. Pri týchto testoch budeme merať priemernú dobu na vloženie jedného znaku, ako aj celkový čas na splnenie testu. Texty, ktoré budeme používať na prepisovanie :

- Anglický text: The quick brown fox jumps over the lazy dog. Triple jumper Phillips Idowu former gymnast Nadia Comaneci exbasketball star John Amaechi and pop star Dizzee Rascal are carrying the flame on its first full day in London.
- Slovenský text : Najvyhľadávanejšia trasa na túry je Tatranská magistrála. Popri relatívne nenáročných úsekoch sú to výstupy na Rysy alebo prechod cez Prielom či Priečne sedlo. Na niektoré štíty je lepšie ísť v sprievode horského vodcu.
- Český text : Léto je v plném proudu, v Tatrách ale teprve vše začíná v sedlech roztávají poslední zbytky sněhu a horské louky pokrývají pestrobarevné květiny. Zveme vás k výletu do části Tater, ležící na samém východním okraji tohoto nejvyššího pohoří Slovenska.

Text pre našu metódu a pre virtuálnu klávesnicu neobsahoval diakritiku. Text pre metódu Tablet PC Input Panel obsahoval diakritiku nie pre znevýhodnenie tejto metódy, ale pre to, že táto metóda umožnuje vkladať len slová vybraného jazyka, čo slová bez diakrityky nie sú.

Program na testovanie sme vyvinuli špeciálne na tieto účely. Skladá sa z dvoch boxov na zadávanie textu. Zatiaľ čo vrchný box obsahuje predlohu, do spodného sa predloha prepisuje. Ako výstup dostaneme výsledok s testovanými hodnotami: dĺžka celého testu, priemerná doba vloženia jedného znaku, maximálna doba vloženia jedného znaku, minimálna doba vloženia jedného znaku.

6.3 Subjektívne metódy testovania

Pri subjektívnej metóde testovania sa zameriame na použitie programu pri bežných činnostiach. Na konci testu dostane testovaný subjekt dotazník. Dotazník pozostával z otázok, ktoré sa týkali testovaného subjektu a otázok na vlastnosti testovaných metód vkladania textu. Osobné otázky boli : vek, pohlavie, počítačové vzdelanie, rodný jazyk a to, či niekedy testovaný subjekt aktívne používal zariadenie Palm, prípadne Tablet PC alebo zariadenie jemu podobné. Na počítačové vzdelanie subjekt odpovedal číslom od 1 do 5. Číslo 1 subjekt zadal, ak seba považuje za neskúseného užívateľa. Číslo 5 zadal, ak sa testovaný subjekt považoval za experta. Na otázky ohľadne vlastností testovaných metód odpovedal subjekt číslom od 1 do 5. Číslo 1 značilo najviac záporné hodnotenie, naopak 5 značilo najviac kladne hodnotenie. Vlastnosti, na ktoré sme sa pýtali boli : vzhľad, jednoduchosť zadávania textu, jednoduchosť ovládania a celkové hodnotenie.
Test bude pozostávať z nasledujúcich úloh:

- Načítanie 5-tich rôznych www stránok (google.com, facebook.com, bbc.com, last.fm, mff.cuni.cz)
- Prihlásenie sa do web-mailu (voliteľne na facebook/ inú sociálnu sieť)
- Odpísanie na mail (voliteľne napísať ekvivalentný text odpovedi na email do poznámkového bloku)
- Vyhľadanie 5-tich ľubovoľných hesiel v ľubovoľnom vyhľadávači.

Tieto úlohy boli vybrané modelovo podľa toho, čo užívatelia najviac robia s tabletami. Podľa prieskumu spoločnosti Google sú to : hranie hier, vyhľadávanie informácií, e-maily, čítanie správ, sociálne siete, počúvanie hudby a pozeranie videí [18].

6.4 Výsledky testovania

Testovanie prebehlo na piatich subjektoch (Tab. 6.1). Subjekty mali rôzne pohlavie, vek aj počítačové vzdelanie. Každý testovaný subjekt ovládal anglický jazyk na úrovni, ktorá bola potrebná na prepísanie anglického textu. Dva testované subjekty v minulosti používali zariadenie Palm. Informácie o subjektoch sú uvedené v tabuľke 6.1. V stĺpci s menovkou „Subjekt“ je uvedené identifikačné číslo subjektu. V stĺpci „Vek“ je uvedený vek subjektu. V stĺpci „Pohlavie“ je uvedené pohlavie subjektu. V stĺpci „Pc.Vzde“ je uvedené počítačová gramotnosť subjektu. V stĺpci Rodný jazyk“ je uvedený rodný jazyk subjektu. V stĺpci „Grafity“ je slovom Áno označené, ak subjekt v minulosti používal Palm alebo jemu podobné zariadenie.

Testovanie prebehlo nakoniec len na texte v anglickom jazyku. Testovanie metódy Tablet PC Input Panel nebolo možné dokončiť na texte napísanom v slovenskom jazyku. Jedine metóda Tablet PC Input Panel sa opiera o lexikálnu analýzu jazyka a keďže pri ostatných metódach bol prepisovaný text bez diakritiky , môžeme považovať test na anglickom texte za ekvivalentný testu v rodnom jazyku.

Objektívne a subjektívne výsledky sú uvedené v tabuľkách 6.2 – 6.5. Pre každý výsledok je štruktúra objektívnych výsledkov rovnaká :

- Stĺpec „Subjekt“ označuje identifikačné číslo testovaného subjektu.
- Stĺpec „Priemer“ uvádza priemerný čas vloženia jedného znaku v milisekundách.
- Stĺpec „Min“ uvádza hodnotu najkratšieho času potrebného na vloženie jedného znaku v milisekundách.
- Stĺpec „Max“ uvádza hodnotu najdlhšieho času potrebného na vloženie jedného znaku v milisekundách.
- Stĺpec „Celkovo“ uvádza celkové trvanie testu v milisekundách.
- Stĺpec „Počet chýb“ uvádza počet chýb ktorých sa subjekt pri prepisovaní dopustil.

V subjektívnych výsledkoch je postupne v stĺpcoch uvedené hodnotenie vzhľadu , jednoduchosti ovládania a celkové hodnotenie.

Hardwarová klávesnica (Tab. 6.2) sa ukázala ako najrýchlejší spôsob na vkladanie textu. Subjekty zvyknuté na písanie 10-timi prstami mali výrazne lepšie výsledky, najlepšie časy a aj najmenší počet chýb. Subjektívne má klávesnica dobré rozloženie a je dostatočne veľká. Negatíve na nej bolo neprítomnosť touchpadu. Touchpoint sa niektorým subjektom používal nepohodlne.

Virtuálna klávesnica (Tab. 6.3) bola najjednoduchšia na vysvetlenie, aj na ovládanie. Všetky subjekty mali skúsenosti s obdobným spôsobom zadávania textu na tabletoch alebo dotykových mobilných telefónov. Presnosť virtuálnej klávesnice bola najvyššia. Rýchlosť pri simulovanom bežnom používaní bola taktiež priemerne najlepšia . Subjektívne vadilo nemožnosť si ponechať virtuálnu klávesnicu na jednom mieste na displeji a neprítomnosť číslic priamo. Dostupné boli až v podmenu klávesnice. Viacero subjektov sa vyjadrilo, že prstami by na nej písali podstatne rýchlejšie.

Tablet Input Panel (Tab. 6.4) dopadol vynikajúco v anglickom jazykoch, ktoré podporoval. Zvládal aj veľmi špecifické štýly písania. Horšie je to s časmi vstupov, ktoré pre vlastnosti analýzy textu sú v niektorých prípadoch pridlhé. Systém je na vysvetlenie jednoduchý. Výsledky skúsenejších subjektov boli veľmi podobné výsledkom nováčikov. Opačný prípad je pri zadávaní textu v slovenčine. Test nebolo možné dokončiť pre nemožnosť písať sloveské slová aj napriek tomu, že vstupný rukopis bol úhľadný a jasne čitelný.

Nami implementované (Tab. 6.5) riešenie má rôznorodé výsledky. Je najzložitejšie na vysvetlenia a zaberie netriviálny čas na naučenie štýlu písania testovacieho subjektu. Pri prvých použitiach má najvyššiu chybovosť a časovú náročnosť vzhľadom na znak. Pri dlhšom používaní a dostatočnom naučení všetkých možných spôsobov písania sa začnú jeho výsledky zlepšovať. Užívateľom vadila nutnosť vyberať si program, do ktorého sa bude prenášať vstup. Páčilo sa grafické spracovanie, ktoré zapadlo do prostredia operačného systému.
Špecifické sú testované subjekty, ktoré v minulosti prišli do styku, respektívne aktívne používali systém na vkladanie znakov Graffity alebo jemu podobný. Na rozdiel od bežných subjektov svoj rukopis prispôsobili na základe tejto skúsenosti na čo najmenej ťahový. Kým bežný testovací subjekt písal písmeno H až tromi ťahmi, subjekt ktorý používal systém Graffity v minulosti na to potreboval jeden ťah. Ich rukopis sa veľmi podobal znakovej sade použitej v systéme grafity.

6.4.1 Vyhodnotenie výsledkov

Riešenia na vstup textu dostupné na operačnom systéme Windows 8 sú prepracované a veľmi funkčné, ale len pre jazyky, ktoré sú podporované. Medzi ne nanešťastie ešte stále nepatrí slovenčina.

Nami implementovaná metóda je použiteľná pre špecifických používateľov, ktorí niekedy v minulosti používali metódu Graffity. Vplyv dlhodobejšieho používania rádovo v dĺžke týždňov sa nám nepodarilo pre nerozšírenosť platformy otestovať. Celkom uspokojívé výsledky sme dosiahli najmä medzi pokročilými užívatelmi. Pre rozšírenie platformy tablet PC najmä medzi nimi môžeme považovať našu metódu za uspokojivú. Pokročilý užívatelia možu nájsť jej praktické uplatnenie.

Informácie o testovaných subjektoch

Subjekt	Vek	Pohlavie	Pc.Vzde	Rodný jazyk	Grafity
1	22	muž	4	Slovenčina	Áno
2	12	žena	2	Slovenčina	Nie
3	45	muž	4	Slovenčina	Nie
4	23	muž	3	Čeština	Áno
5	64	muž	3	Slovenčina	Nie

Tab. 6.1 Popis testovaných subjektov

Hardwarová klávesnica

	Objektívne výsledky					Subjektívne výsledky		
Subjekt	Priemer	Min	Max	Celkovo	Počet chýb	Vzhľad	Jednoduchosť	Celkovo
1	306,36	43,00	5 766,33	66 480,80	16,00	5	5	5
2	373,76	52,46	7 034,92	81 106,58	4,00	3	3	4
3	336,99	47,30	6 342,96	73 128,88	15,00	4	5	4
4	349,25	49,02	6 573,62	75 788,11	8,00	4	4	4
5	263,47	36,98	4 959,04	57 173,49	12,00	4	4	4
Priemer	325,97	45,75	6 135,37	70 735,57	11,00	4	4,2	4,2

Tab. 6.2 Výsledky testov pre hardwarovú klávesnicu

Virtuálna klávesnica

	Objektívne výsledky					Subjektívne výsledky		
Subjekt	Priemer	Min	Max	Celkovo	Počet chýb	Vzhľad	Jednoduchosť	Celkovo
1	829,87	167,52	14 550,50	180 088,38	26,00	4	5	4
2	738,59	149,09	12 949,94	160 278,66	18,00	4	4	4
3	780,08	157,47	13 677,47	169 283,07	25,00	3	4	5
4	987,55	199,35	17 315,09	214 305,17	14,00	2	4	3
5	1 020,74	206,05	17 897,11	221 508,70	22,00	4	3	4
Priemer	871,37	175,90	15 278,02	189 092,80	21,00	3,4	4	4

Tab. 6.3 Výsledky testov pre virtuálnu klávesnicu

Tablet PC Input Panel

	Objektívne výsledky					Subjektívne výsledky		
Subjekt	Priemer	Min	Max	Celkovo	Počet chýb	Vzhľad	Jednoduchosť	Celkovo
1	70,85	65,13	85,33	287 479,34	20,00	2	3	3
2	88,57	81,42	106,66	359 349,18	10,00	2	3	3
3	87,86	80,77	105,80	356 474,38	2,00	3	2	2
4	58,10	53,41	69,97	235 733,06	9,00	3	3	3
5	59,52	54,71	71,67	241 482,65	4,00	2	2	2
Priemer	72,98	67,09	87,89	296 103,72	9,00	2,4	2,6	2,6

Tab. 6.4 Výsledky testov pre Tablet PC Input panel

Naša metóda

	Objektívne výsledky					Subjektívne výsledky		
Subjekt	Priemer	Min	Max	Celkovo	Počet chýb	Vzhľad	Jednoduchosť	Celkovo
1	2 886,88	15,00	27 789,85	346 455,00	20,00	5	4	4
2	2 886,88	15,00	27 789,85	346 455,00	16,00	4	2	2
3	2 713,67	14,10	26 122,46	325 667,70	12,00	4	1	2
4	3 175,57	16,50	30 568,83	381 100,50	2,00	5	4	4
5	2 338,37	12,15	22 509,78	280 628,55	3,00	4	1	2
Priemer	2 800,27	14,55	26 956,15	336 061,35	10,60	4,4	2,4	2,8

Tab. 6.5 Výsledky testov pre našu metódu

Záver

V priebehu práce sme vyvinuli a implementovali postup na zadávanie textu pomocou rukou písaných znakov.
Na dosiahnutie tohto bol vyvinutý postup na zjednodušenie rukou písaného textu, pozostávajúceho aj z viacerých ťahov pre znak. Znak sme úspešne previedli podľa myšlienky prevzatej z kaligrafie a pomocou algoritmov na rozoznávanie rohov a aproximáciu kriviek na vektorovú sekvenciu čísiel. Pomocou pozičnej analýzy ťahov sme vytvorili pozičnú sekvenciu čísiel. Na základe týchto dvoch sekvencií sme schopní priradiť k nakreslenému útvaru odpovedajúci znak.

Nanešťastie, naša metóda nie je vhodná pre každého. Pre začínajúcich užívateľov platformy tablet PC bola nepohodlná a od používania ich odradzovalo implementačné obmedzenie výberu programu pre vstup. Výsledný program ktorý sme implementovali slúžil na experimentálne overenie funkčnosti nášno riešenia. Pre reálne používanie by naše riešenie bolo plne integrované do operačného systému, teda bez nutnosti vyberania aplikácie pre vstup textu. Užívatelia zbehlejší s touto platformou a jej predchodcami využili naše riešenie ako návrat formy zadávania textu pomocou metódy Graffity. Naše riešenie im pridáva možnosť predefinovať si niektoré znaky pre ich najjednoduchšie použitie.

S novým operačným systémom Windows 8 bude počet nových užívateľov na tejto platforme stúpať. Prezentované riešenie má medzery práve s novými užívateľmi. Do budúcnosti by sme chceli preklenúť priepasť medzi novým užívateľom, ktorý nemal ešte s podobnou metódou vstupu skúsenosti. Užívatelia boli skeptickí aj pri virtuálnych klávesniciach pričom teraz je to najobľúbenejšia, najpresnejšia a najrýchlejšia metóda zadávania textu.

Novým užívateľom by sme pribĺiž našu metódu vkladania textu vytvorením intenzívneho tutoriálu, ktorý by v krátkom čase naučil užívateľa na základe jeho rukopisu jemu prispôsobený zjednodušený rukopis. Program ktorý by obsahoval tutoriál by zjednodušil rukopis do čo najmenšieho počtu ťahov, ktoré pripadajú na znak. Takto zjednodušené písmo by zahrňovalo užívateľove špecifické črty jeho rukopisu, takže by neprišiel o pohodlie a istotu pri písaní. Znakovo orientované zadávanie textu istotne má miesto medzi ostatnými metódami zadávania textu do počítačov druhu tablet PC.

V budúcnosti by sme chceli pridať možnosť písania diakritiky. Možné implementačné riešenia sú pri zachovaní vlastností gestom vyvolané podmenu, ktoré by pridalo diakritiku na nasledujúci znak, alebo možnosť do oblasti okolo miesta, kde bol napísaný posledný znak, napísať požadovanú diakritiku, ktorá by bola pridaná na posledný znak.

Zoznam použitej literatury

[1] HUURDEMAN, Anton A. *The worldwide history of telecommunications.*. New York: J. Wiley, c2003, s. 148-153. ISBN 978-0-471-20505-0.

[2] BUTTER, Andrea a David POGUE. *Piloting Palm: the inside story of Palm, Handspring, and the birth of the billion-dollar handheld industry.* New York: John Wiley, c2002, xiii, 353 p., [8] p. of plates. ISBN 04-710-8965-6.

[3] O'GRADY, Jason D. *Apple Inc.* Westport, Conn.: Greenwood Press, 2009, s. 35-59. Corporations that changed the world. ISBN 978-0-313-36244-6.

[4] ATKINSON, Paul. *A Bitter Pill to Swallow: The Rise and Fall of the Tablet Computer [online].* Massachusetts, 2008 [cit. 2012-07-30]. Dostupné z: http://www.mitpressjournals.org/doi/pdf/10.1162/desi.2008.24.4.3. Clanok. Massachusetts Institute of Technology.

[5] JARRETT, Rob a Philip SU. *Building Tablet PC applications.* Redmond, Wash.: Microsoft Press, 2002, s. 10-22. ISBN 0-7356-1723-6.

[6] BAKER, Arthur. *Arthur Baker's Historic calligraphic alphabets.*. New York: Dover Publications, 1980, 89 p. ISBN 04-862-4054-1.

[7] LEVIN, Michael. *CellWriter: Grid-Entry Handwriting Recognition.* Minnesota, 2007. Dostupné z: http://risujin.org/cellwriter/cellwriter.pdf. Diplomová práca. University of Minnesota UROP. Vedoucí práce Prof. Nikolaos Papanikolopoulos.

[8] FINK, Gernot A, *Markov models for pattern recognition: from theory to applications.* London: Springer, c2008, xii, 248 s. ISBN 978-3-540-71766-9.

[9] ATTNEVE , Fred. *Some informational aspect of visual perception* Psycho. Rev. vol. 61. no 3. s. 183-193, 1954

[10] ROSENFELD, Adam a Peter Johnston *Angle detection on digital curves* IEEE Trans. Comput. vol. C-22 s. 875-878. September 1973

[11] ROSENFELD, Adam a Tomas WESKA J.S *An improved method of angle detection on digital curves* IEEE Trans. Comput. vol. C-24 s. 940-941. September 1975

[12] CHETVERIKOV, Dmitry a Zsolt SZABO. *A simple and Efficient Algorithm for Detection of High Curvature Points in Planar Curves.* Budapest. Computer and Automation Research Institute.

[13] HECKBERT, Paul S. a Michael GARLAND *Survey of Polygonal Surface Simplification Algorithms.* Pittsburgh, PA 15213, 1 Máj 1997. School of Computer Science Carnegie Mellon University.

[14] FREEMAN, Herbert a Larry DAVIS. *A Corner-Finding Algorithm for Chain-Coded Curves. IEEE Transactions on Computers.* 1977, C-26, č. 3, s. 297-303. ISSN 0018-9340. DOI: 10.1109/TC.1977.1674825. Dostupné z:

http://ieeexplore.ieee.org
/lpdocs/epic03/wrapper.htm?arnumber=1674825

[15] PLAMONDON, Réjean a Sargur N. SRIHARI. *Online and off-line handwriting recognition: a comprehensive survey,* Transactions on Pattern Analysys and Machine Intelligence, vol. 22, Január 2000

[16] MACKENZIE, Scott a Larry CHANG. *A performance comparison of two handwriting recognizers,* Interacting with Computers, vol. 11, pp. 283–297, Január 1999.

[17] PITTMAN, James A. *Handwriting Recognition: Tablet PC Text Input. Computer.* 2007, roč. 40, č. 9, s. 49-54. ISSN 0018-9162. DOI: 10.1109/MC.2007.314. Dostupné z: http://ieeexplore.ieee.org
/lpdocs/epic03/wrapper.htm?arnumber=4302613

[18] *Understanding Tablet Device Users.* Thinkwithgoogle [online]. [cit. 2012-07-20]. Dostupné z: http://www.thinkwithgoogle.com
/insights/library/studies/understanding-tablet-device-users/

[19] *Moon type.* In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-08-02]. Dostupné z: http://en.wikipedia.org/wiki/Moon_type

Prílohy

Obsah CD-ROM

Obsah sprievodného CD-ROM je nasledovný:

- `Práca` — táto práca (vo formátoch PDF a PS) a jej LaTeX-ový kód v archíve ZIP.
- `Program.zip` — obsahuje skompilovateľné zdrojové súbory Programu v jazyku C# (obsahuje aj knižnicu .NET 4.0 potrebnú na spustenie programu).
- `Bakalárska práca-programátorská dokumentácia` — Programátorská dokumentácia
- `Kontakt.txt` — kontaktné údaje autora práce.